O *Sinclair QL* oferece imensas oportunidades aos jovens para criarem aventuras excitantes e exigentes. A enorme memória e as excelentes possibilidades gráficas da máquina são exploradas neste livro, cujos autores demonstram ao leitor como escrever a sua própria aventura, utilizando o programa especialmente concebido «Gerador» *(Generator)*.

O «Gerador» é um construtor de aventuras que pode ser usado para criar aventuras exclusivas, de acordo com o desejo de cada um, permitindo tirarem-se as maiores vantagens de todas as possibilidades do *Sinclair QL*. O «Gerador» é acompanhado de um programa de processamento de dados, que pode ser usado em conjunto com o «Gerador», para fornecer os pormenores da própria aventura.

Este livro é um precioso guia, quer para os entusiastas das aventuras quer para os que estão ávidos de explorar esta excitante área dos microcomputadores, utilizando um dos mais poderosos «micro» domésticos.

Tony Bridge é colaborador permanente do *Popular Computing Weekly* e do *MicroAdventurer*.

O Dr. Richard Williams orienta diversos cursos de computadores e de inteligência artificial. É o autor de inúmeros artigos e livros sobre informática.

ARTE DE VIVER
107

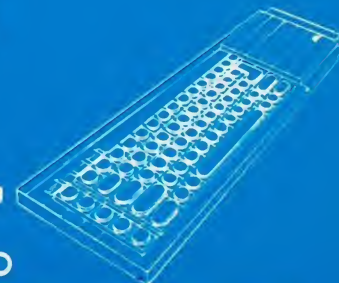
13310786 1

Tony Bridge e Richard Williams

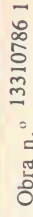
JOGOS DE AVENTURAS PARA O Sinclair QL

O manual do microaventureiro

PUBLICAÇÕES EUROPA-AMÉRICA

AVISO DE LANÇAMENTO

PARA O LIVREIRO DESTACAR DO LIVRO NO ACTO DA VENDA

Obra n.º 13310786 1

Título: *JOGOS DE AVENTURAS PARA O SINCLAIR — UM GUIA PARA JOGOS DE AVENTURAS EM MICROCOMPUTADOR*

Autor: Tony Bride e Richard Williams

Colecção: «Arte de Viver»

Preço:.................. $ .....

☐ 1.º Fornecimento

☐ Reposição

Factura de fornecimento

N.º .... Data .../.../...

Código do cliente

GUARDE ESTE TALÃO

O LIVREIRO DEVE GUARDAR ESTE TALÃO, APÓS A VENDA DO LIVRO, PARA SEU CONTROLO DE VENDAS E REPOSIÇÃO DE NOVO EXEMPLAR.

Este livro só poderá ser devolvido se fornecido em SERVIÇO DE ASSINATURA DE NOVIDADES e de acordo com as cláusulas 12.ª a 17.ª do Contrato de Agência.

Mod. 12.293 — Impresso por Gráfica Europam, L.da

JOGOS DE AVENTURAS PARA O SINCLAIR QL

O manual do microaventureiro

Obras publicadas nesta colecção:

1 — *111 Receitas com Ovos*, Etelvina Lopes de Almeida
2 — *O Livro do Casal*, Pierre-Marie Brémond
3 — *Aprenda a Fotografar*, Antoine Desilets
4 — *Guia da Interpretação dos Sonhos*, Louis Stanké
5 — *A Arte de bem Receber*, Marguerite du Coffre
6 — *Guia do Comportamento Sexual*, Dubois-Caballero
7 — *Como Reparar Avarias na Estrada — Manual de Todo o Automobilista*, Miguel de Castro Vicente
8 — *Guia Prático e Completo da Costura*, Lise Chartier
9 — *Guia Íntimo das Relações Sexuais*, Pierre Valinieff
10 — *Guia dos Jovens — A Vida e o Amor*, Dr. Benjamin Spock
11 — *111 Receitas de Tapas e Entradas*, Etelvina Lopes de Almeida
12 — *Guia da Futura Mãe durante a Gravidez*, Dr. José M.ª Carrera
13 — *Como Suprimir as Suas Dores com a Simples Pressão de Um Dedo*, Dr. Roger Dalet
14 — *O Livro das Boas Maneiras*, Marcelle Fortin-Jacques
15 — *111 Receitas de Frango*, Jacky Davin
16 — *Doenças Transmitidas pelas Relações Sexuais*, Dr. Lionel Gendron
17 — *Hatha-Yoga*, Suzanne Piuze
18 — *Os Segredos do Amor Táctil*, A. Vignati e O. Caballero
19 — *Como Socorrer o Seu Filho*, Marie Hermand
20 — *Métodos Anticonceptivos e Planeamento Familiar*, Santiago Dexeus e Margarita Riviere
21 — *A Técnica da Fotografia*, Antoine Desilets
22 — *Amor, Sexo e Astrologia*, Teri King
23 — *Como Vencer a Timidez*, François Suzzarini
24 — *111 Receitas de Coelho*, Anne Vernon
25 — *Os Remédios da Avozinha*, Barbara Kamir
26 — *A Mulher depois dos 40 Anos*, Santiago Dexeus e Teresa Pámies
27 — *Viver bem depois dos 50 Anos*, Dr. Hugues Destrem
28 — *Conservas, Compotas e Xaropes*, Maria Emília Abreu Semedo
29 — *Como Proteger a Saúde e a Beleza com a Simples Pressão de Um Dedo*, Dr. Roger Dalet
30 — *111 Receitas para Emagrecer*, Dr. Jean-Paul Ostigny
31 — *Eu... Tu... e os Outros*, Anna Boyer e Isabelle Nicolas
32 — *O Rosto, Espelho do Carácter*, Louis Stanké
33 — *O Seu Aquário de Peixes Tropicais*, Brian Ward
34 — *Pílula — A Solução Mortal*, Dr. Dominique Chatain
35 — *O Seu Futuro nas Cartas*, Louis Stanké
36 — *111 Refeições Naturistas*, Maria Cândida de Albuquerque Cardoso
37 — *A Bíblia do Bridge*, Claude Derwy
38 — *A Congelação dos Alimentos*, Pamela Dotter
39 — *A Celulite*, Gerald J. Leonard
40 — *Guia Sexual da Moça Moderna*, Wardel B. Pomeroy
41 — *101 Conselhos aos Diabéticos*, Prof. Georges Tchobroutsky
42 — *A Beleza pela Saúde*, Dr. Pierre Fournier
43 — *Plantas de Interior*, Brian Ward e Tom Wellsted
44 — *111 Receitas de Cozinha Africana*, Maria de Lourdes Chantre
45 — *Saber Maquilhar-Se*, Josette Ghedin
46 — *111 Receitas de Massas*, Anne Vernon
47 — *Alimentação Natural*, José Lyon de Castro
48 — *A Mulher e o Sexo*, Dr. Lionel Gendron
49 — *A Menopausa*, Dr. Lionel Gendron
50 — *111 Receitas de Arroz*, Déda Frachon
51 — *Trate o Seu Cão, o Seu Gato, os Seus Pássaros com a Simples Pressão de Um Dedo*, Roger Dalet
52 — *A Chave da Longevidade*, Dr. Hugues Destrem
53 — *Tudo sobre Acupunctura*, Dr. Jean Vibes
54 — *101 Respostas sobre a Depressão*, Dr.ª Marie Claude Navikoff e Dr. Jean Pierre Olié
55 — *111 Receitas para Panela de Pressão*, Janet Warren
56 — *Como Vencer as Enxaquecas*, Dr. Claude Loisy e Dr. Sidney Pélage
57 — *Como Viajar de Avião sem Ter Medo*, Afra Botteri/Cécile Gateff
58 — *Tempo Que Mata, Tempo Que Cura*, Dr. Fernand Attali
59 — *Como Manter a Virilidade*, Paul Stanley
60 — *A Alimentação da Criança*, Louise Lambert-Lagacé
61 — *O Sexo e o Amor no Casamento*, Bernard Delon e Germaine Lanoë
62 — *Conheça-Se a Si Próprio — I*
63 — *Conheça-Se a Si Próprio — II*
64 — *Ténis Prático — Técnica — Conselhos — Campos*, Christian Collin
65 — *101 Segredos da Medicina Natural*, Dr. Péron-Antret
66 — *Manual de Protecção contra o Crime*, Ira A. Lipman
67 — *111 Receitas de Caça*, Ana Isabel de Castro
68 — *Manual Médico da Família*, Dr. David Kellett Carding
69 — *A Cozinha Astrológica*, Marie Gebert e Monique Maine
70 — *Doenças de Cães e Gatos Transmissíveis a Crianças*, Silva Leitão
71 — *Manual de Sobrevivência na Situação de Guerra Nuclear — Como Viver durante e após um Ataque Nuclear*, Barry Popkess
72 — *Conheça os Computadores*, John Shelley
73 — *Como Tratar o Seu Filho com a Simples Pressão de Um Dedo*, Dr. Tan Poh Choon
74 — *Tudo sobre Astrologia*, H.-M. de Campigny
75 — *Horóscopos Árabes*, Paula Delsol
76 — *Horóscopos Chineses*, Paula Delsol
77 — *Aventuras com o Spectrum*, Tony Bridge e Roy Carnell
78 — *Enciclopédia dos Pontos Que Curam*, Dr. Roger Dalet
79 — *A Dianética*, L. Ron Hubbard
80 — *Auto-Análise*, L. Ron Hubbard
81 — *Como Vencer no Trabalho e na Vida*, L. Ron Hubbard
82 — *Como Planear e Construir a Sua Lareira*, Margaret e Wilbur F. Eastman Jr.
83 — *As Previsões Astrológicas para 1985*, Catherine Aubier
84 — *Faça Você Mesmo — I — Alvenaria, Telhados, Carpintaria*
85 — *Faça Você Mesmo — II — Electricidade, Canalização, Pintura, Vidraria*
86 — *Faça Você Mesmo — III — Revestimentos, Isolamentos, Refrigeração*
87 — *8 Exercícios para Um Corpo Perfeito*, Sheri Blair
88 — *Guia Prático da Sorte*, Cécile Donner e Jean-Luc Caradeau
89 — *Programação Prática para o Spectrum em Linguagem Máquina*, Steve Webb
90 — *Aplicações Domésticas no seu Microcomputador*, Mike Grace
91 — *Como Fazer Amor com a Simples Pressão de Um Dedo... e não só*, Hsuan Tsai Su-Nu
92 — *111 Receitas de Cozinha Indiana*
93 — *Como Fazer Amor com Um Homem*, Régine Dumay
94 — *Terapêutica Biológica*, Adriano de Oliveira
95 — *Inteligência Artificial no Spectrum*, Keith e Steven Brain
96 — *111 Receitas de Cozinha Chinesa*
97 — *O Spectrum Funcional*, David Lawrence
98 — *Domine o Seu ZX Microdrive*, Andrew Pennell
99 — *Desenvolvimento de Aplicações no Sinclair QL — Ideias práticas para utilizações domésticas e em negócios*, Mike Grace
100 — *Truques de Ilusionismo*, Araújo
101 — *Receitas de Refeições para Bebés, Crianças e Jovens — Crescer com Saúde*, Catherine Lewis
102 — *Comer Bem e Barato com Saúde — Dieta Para Evitar o Cancro*, Carmel Berman Reingold
103 — *A Inteligência Artificial no Sinclair QL — Faça o Seu Micro Pensar*, Keith e Steven Brain
104 — *Manual de Defesa Pessoal*, Prof. J. A. Fonseca Gaspar
105 — *Ervas — Aplicações Culinárias Decorativas e Cosméticas*, Jack Harvey
106 — *A Conservação de Alimentos*, Pamela Dotter
107 — *Jogos de Aventuras para o Sinclair QL — O manual do microaventureiro*

Tony Bridge e Richard Williams

JOGOS DE AVENTURAS PARA O Sinclair QL

O manual do microaventureiro

ARTE DE VIVER®

PUBLICAÇÕES EUROPA-AMÉRICA

Título original: Sinclair QL Adventures

Tradução de Maria do Rosário Prata Ferreira dos Santos

Capa: estúdios P. E. A.

Editor: Francisco Lyon de Castro

PUBLICAÇÕES EUROPA-AMÉRICA, LDA.
Apartado 8
2726 MEM MARTINS CODEX
PORTUGAL

Edição n.º 133107/4063

Execução técnica:
Gráfica Europam, Lda.,
Mira-Sintra — Mem Martins

Índice

AGRADECIMENTOS

Agradece-se aos serviços dos Correios por não terem perdido nenhuma das nossas preciosas *cartridges;* à Mobil por não ter levantado o preço da gasolina enquanto escrevemos este livro; à Fiat Motors inglesa os transportes de qualidade; a Sascha Kerry e Lucy por numerosas chávenas de chá e a Jilly pelas decorações exclusivas.

Introdução

Os jogos de aventuras já não são meros divertimentos ilícitos de entediados programadores de computador, utilizando, fora das horas de serviço, computadores principais caríssimos e tempo de computador muito dispendioso para combater dragões e serpentes. Estes jogos transformaram-se num grande negócio e, com a actual explosão de vendas de computadores pessoais, tornaram-se os favoritos de milhões de entusiastas dos computadores por esse mundo fora.

De facto, nos últimos meses, enquanto este livro estava a ser escrito, notámos um desvio do interesse votado aos jogos de *arcade* na direcção dos jogos de aventuras. A que causas se deverá este facto? Pensamos que os jogos de *arcade* atingiram um plano limite. Estes jogos começaram, nos tempos do PET/Nascom, com *hardware* muito dispendioso, passando o programa *Space Invaders*. Os *Space Invaders* eram X e W, mas a falta de apoio visual não diminuía o interesse pelo jogo: era suficiente esta pálida imitação do clássico caça-níquel, além de que não era necessário arranjar uma moeda de cada vez que se queria jogar — ignorando comodamente, é claro, a elevada verba necessária para a compra do aparelho! Isto foi o início. Os *Space Invaders* dominaram durante anos, até que, finalmente, uma ou duas almas intrépidas, entre as quais Clive Sinclair no Reino Unido, colocaram o computador ao alcance dos adolescentes. Isto levou ao aparecimento de novos jogos destinados a esta delicada audiência.

Primeiro apareceram os sucedâneos dos *Space Invaders,* tais como *Galaxians, Gorf,* etc.; depois foi a vez de *Pacman,* com inúmeras versões e variantes. Cada novo nível de perícia alcançado pe-

los pioneiros facilitava a tarefa da multidão de programadores que se seguiu. Os jogos *Pacman, Kong* e *Invader* continuam a ser uma inestimável ajuda para os programadores principiantes, mas apareceram também em cena alguns jogos novos e muito originais.

De qualquer modo, foi alcançado um plano limite, por muito alto que esse limite seja. No nível actual da tecnologia de computadores, não é possível fazer mais que inventar novas maneiras de dar cabo de extraterrestres, atravessar labirintos, ou subir escadas. Cada novo aparelho que é anunciado introduz novas formas de facilitar a programação de efeitos nos nossos jogos de *arcade,* mas esses efeitos mantêm-se aproximadamente os mesmos. Fizeram-se várias tentativas de utilização de novas técnicas — na altura em que este livro está a ser escrito, os jogos tridimensionais (alguns dos quais são mais bem sucedidos que outros) suscitam um certo interesse, e os saudosos equipamentos *Imagine* deixaram no ar a ideia dos megajogos, seja o que for que eles pudessem ser; mas, no fim de contas, só um maior realismo pode levar os jogos de *arcade* a dar um salto em frente.

O que os jogadores procuram num jogo de *arcade* de tiroteio é precisamente o realismo. O perigo, independentemente de ser sentido por interposta pessoa, deve parecer real: a adrenalina tem de subir. Porém, o actual estado de desenvolvimento do *hardware* não permite proporcionar ao jogador mais que um envolvimento simbólico na acção que se desenrola no visor.

Há uma grande diferença entre jogar um jogo numa sala de jogos de um centro comercial, com a barulheira que lhe está geralmente associada (gritos, assobios...), e jogar o mesmo jogo num microcomputador. Os jogos de *arcade* mantêm-se, em média, apenas durante alguns meses (os grande êxitos duram muito mais, é claro), mas o negócio dos jogos é um viveiro de invenções. Existem hoje em dia máquinas que usam discos vídeo para criar um ambiente realista à volta do jogador, que pode estar sentado numa cabina escura, frente ao visor, que pode envolvê-lo completamente, embrenhando-o cada vez mais na acção. As ligações em rede também começam a aparecer, permitindo a dois ou mais jogadores jogarem uns contra os outros em diferentes aparelhos. No futuro próximo podemos esperar o uso de holografia, dando ao jogador uma sensação de realidade tridimensional. E depois disso, quem sabe? Talvez um dia um jogo possa vir a proporcionar sensações tácteis ou olfactivas. No fim de contas, o som e as impressões visuais têm um grande impacte nos actuais jogos de *arcade* e, assim sendo, que é que falta?

A maior parte destas técnicas — com a possível excepção das ligações em rede, que parecem ter sido feitas por medida para os microcomputadores — é demasiado dispendiosa e ocupa demasiado espaço para que valha a pena implementá-la nos computadores pessoais.

O próximo passo em direcção ao realismo nos jogos para computadores pessoais só pode ser uma espécie de laço mental entre o aparelho e a mente humana. Isso pertencerá, obviamente, a um futuro longínquo, e, por muito excitante que possa ser imaginá-lo agora, o contacto entre cérebros deve trazer consigo coisas demasiado negras para que possam ser previstas.

Existirá outro tipo de jogos para computador em que se possa dar largas à imaginação e criar um mundo novo para o jogador? Existe, sim, e o leitor já adivinhou com certeza qual: trata-se, evidentemente, dos jogos de aventuras.

Os jogos de aventuras progrediram de forma muito semelhante aos jogos de *arcade:* mantiveram-se na sombra durante muito tempo até a actual geração de computadores pessoais os ter arrastado para a luz. De então até agora, os jogos de aventuras ainda não atingiram um plano limite, como aconteceu com os jogos de *arcade,* pois os seus cenários fazem largamente parte da imaginação colectiva do autor e do jogador.

QUAIS SÃO OS INGREDIENTES DE UM BOM JOGO DE AVENTURAS?

O primeiro jogo de aventuras, *Colossal Cave,* foi escrito para um computador principal. Nesse tempo os computadores ainda não dispunahm de VDU (*visual display units* — visores). Em vez disso, os resultados eram obtidos através de uma impressora ligada ao computador. O jogo aparecia sob a forma de texto expelido pela impressora. Isto levou a que se originasse entre os adeptos de aventuras em computador o mito de que o texto é o melhor, o que não é necessariamente verdadeiro.

As imagens de alta resolução *podem* acrescentar imensa atmosfera e, ocasionalmente, fornecer uma pista pouco habitual, mas o preço a pagar é um forte dispêndio da memória disponível. Infelizmente, na maior parte dos microcomputadores pessoais, a memória é preciosa. Em geral, uma aventura escrita em linguagem máquina pode, razoavelmente, incluir até cem locais, com um par de linhas de texto e todos os objectos e quebra-cabeças com eles relacionados. Um menor número de locais permitirá descrições mais pormenorizadas, ou um aumento do número de objectos ou de quebra-cabeças. Se se acrescentar apoio visual, esse número pode bem ser reduzido em dois terços.

Um compromisso possível é ter a aventura gravada em disco, podendo os dados ser chamados à medida que o jogador vai chegando a cada local. É assim que funciona a maior parte das aventuras americanas, ainda que, geralmente, existam versões reduzidas disponíveis em fita. Outra forma de criar uma aventura maior, que seria, de outro modo, possível com o uso de fita, é ter duas ou mais partes ou «capítulos» de um jogo, sendo cada uma delas introduzida sob a forma de fita à medida que for mecessário. No entanto, este método é bastante pouco flexível e não é uma solução tão boa quanto o disco.

Felizmente, a memória vai-se tornando mais barata à medida que os meses passam, e agora, com o QL, temos suficiente RAM (*random access memory* — memória de acesso aleatório) para construir uma aventura bastante grande. Apesar disso, existe uma espécie de lei de Parkinson da memória de computador, segundo a qual as necessidades presentes se expandem sempre de modo a ocupar toda a memória disponível. Quando o *ZX81,* de 1 k foi lançado, faziam-se maravilhas com a sua reduzida RAM e todos esperavam ansiosamente pela expansão de 16K, dizendo que era perfeitamente suficiente e que ninguém iria utilizar mais de 16K. É claro que, pouco tempo depois, tudo estava cheio, e começaram a aparecer extensões de 32K. Depois, foi lançado o *Spectrum* e a venda da versão de 48K foi várias vezes superior à do modelo de 16K. Ainda que 128K nos pareça ser mais que suficiente para os nossos objectivos actuais, dentro de alguns meses, à medida que forem aparecendo programas cada vez mais complexos, vai, certamente, parecer-nos exíguo. Existe ainda a possibilidade de utilizar os *microdrives* do *QL* de forma a simular uma aventura baseada em disco. Assim, o código para o programa seria introduzido no *QL,* e uma segunda

cartridge seria então utilizada para introduzir os dados sobre os locais à medida que fosse necessário.

Vimos, portanto, que uma boa aventura pode ser constituída *tanto* apenas por texto *quanto* por texto com apoio visual, e que o seu sucesso não está dependente de qualquer destes formatos. Também é possível a existência de aventuras baseadas puramente em imagens: a este tipo chamamos *arcventures.* Entre os programas bem conhecidos que se enquadram nesta categoria (ainda que muita gente os considere meramente jogos de *arcade*) temos *Jet Set Willy, Manic Miner, Miner 2049'er* e *Jumpman,* além de muitos outros exemplos semelhantes. Ainda que apresentem muitas das características que se encontram nos jogos de *arcade,* estes jogos têm um elemento (de facto, é mesmo o seu elemento principal) de resolução de quebra-cabeças. O espaço não nos permite entrar em pormenores sobre este assunto, mas as *arcventures* são tratadas com mais profundidade em *Atari Adventures,* de Tony Bridge (publicado pela Sunshine Books).

COMO PRATICAR JOGOS DE AVENTURAS

Deitemos uma breve vista de olhos a um texto típico de um jogo de aventuras.

Há duas formas principais de conduzir jogos de aventuras. Na primeira, o jogador dá ao computador instruções acerca do que deve ser feito. Chama-se a este método o «método da marioneta», porque o computador age como se fosse uma marioneta manobrada pelo jogador. Assim, uma interacção típica poderia ser:

Estou num cemitério rodeado por velhas pedras tumulares parcialmente obscurecidas por uma névoa deslizante. Nas sombras posso ver uma forma arrepiante, de olhos brilhantes, vermelho-escuros, que vem avançando em direcção a mim. As saídas evidentes são Norte e Oeste, e ao meu lado encontra-se uma campa aberta. Que é que hei-de fazer?

A resposta pretendida pode ser, no primeiro caso, GO NORTH (vá para norte), ainda que esta resposta possa normal-

mente, em qualquer programa de jeito, ser reduzida a NORTH, ou mesmo a N. Esta é a resposta mais básica: há muitas outras que vão ser requeridas, dependendo das circunstâncias particulares do momento. O jogador pode, por exemplo, ter agarrado numa espada num passo anterior do jogo, e dar agora instruções ao computador para a utilizar atacando a forma arrepiante. Ou mesmo, fazendo nesta altura uma busca (SEARCH) ao cemitério, poderia encontrar um objecto útil junto de um túmulo. Além disso, nenhum aventureiro que se preze deixaria a campa aberta por explorar!

O segundo método de praticar jogos de aventuras em computador é o método da «primeira pessoa», em que é o próprio jogador o protagonista da aventura. Assim:

Você está num cemitério...

Ainda que não haja muita diferença entre este dois processos, tendemos a preferir o método da marioneta, dado que o jogo dá origem a um texto mais personalizado. Falaremos mais pormenorizadamente deste assunto no segundo capítulo.

Como já nos habituámos a esperar, a editora americana Infocom criou e lançou o primeiro jogo de aventuras na «sexta» pessoa, *Suspended,* que é discutido mais adiante.

PRIMEIRA PARTE

História dos jogos de aventuras

CAPÍTULO 1

Origens

Depressa, acendam a luz! Ah, assim é melhor, agora podemos ver até um pouco mais longe na obscuridade. Os ruídos estranhos desapareceram quando a luz se acendeu, e agora está tudo silencioso. Ali à frente há uma abertura na parede da gruta; vamos entrar e ver o que nos espera.

Olá!? Que é isto? Uma vara preta caída no chão húmido da gruta? Vá, toca a apanhá-la. Haverá mais alguma coisa por aqui? Não, é tudo, penetremos mais nas catacumbas. Na próxima gruta está uma gaiola de verga atirada para um canto. Será uma armadilha? Vou tentar empurrá-la com a vara, agitar a vara na direcção dela... Bom, não acontece nada, por isso posso apanhá-la.

Da próxima gruta chega-me o ténue som do canto de um pássaro. Avançando silenciosamente para o interior da gruta podemos ver um pássaro a cantar alegremente sobre uma pedra próxima. A gaiola deve ser para isso: vamos apanhar o bicho! Mas ele voou... Alguma coisa o deve ter assustado. Realmente, a vara tem um aspecto um bocado ameaçador. Vou pousá-la. Óptimo, o pássaro voltou a pousar e canta alegremente, alheado de nós, que nos esgueiramos em direcção a ele com a gaiola. Já está! Apanhámo-lo.

Paremos para agarrar a vara (que vai certamente ser útil mais tarde) e continuemos para a gruta seguinte, iluminando bem o caminho à nossa frente. De repente, surgindo da obscuridade, uma grande sombra ergue-se na nossa frente. Uma enorme serpente verde e coleante fixa-nos com o olhar. Não há outra saída à vista, por isso temos de passar pela fera para podermos prosseguir a nossa demanda. Vamos parar e pensar um bocado.

Talvez agitar a vara resulte. Não, a serpente não parece perturbada por isso. Que tal usar a vara como uma vara de salto? Não dá, a serpente é grande de mais para que seja possível saltar por cima dela. O passarinho continua a cantar alegremente. Espera, talvez seja uma boa ideia dá-lo a comer à serpente. Ela é capaz de ter fome, e isso talvez a mantenha ocupada tempo suficiente para que nós passemos e nos vamos embora rapidamente. Soltemos o pássaro e vejamos o que acontece. Será possível? A serpente, assustada com o adejar das asas, assobia violentamente e desliza para as trevas, deixando-nos livres para continuar a nossa demanda.

Eis um cenário típico de uma aventura computadorizada, na qual a violência explícita não desempenha um papel importante, Porém, existem variantes em que as capacidades de luta e de manejo de armas se sobrepõem ao poder de raciocício. Mais tarde, trataremos de algumas dessas variantes, mas, por agora, vamos definir o que entendemos por aventura.

Para encontrar as origens da aventura computadorizada, temos de tratar primeiro de outro entretenimento pertencente à era anterior ao computador pessoal (lembram-se desses tempos, antes de passarem os serões colados à última versão de *Zaxxon* ou de *Qix?*) Trata-se de um passatempo tão velho quanto o xadrez, e quase tão velho como o mais velho dos passatempos.

JOGOS DE GUERRA

Os jogos de guerra são praticados desde que os chefes tribais conseguiram juntar exércitos de mais de meia dúzia de homens. De facto, até as mais simples manobras de combate necessitam de ser treinadas. Haverá melhor maneira de o fazer que transformar o treino num jogo, incentivando assim o processo?

Foram inventadas algumas versões muito complexas de jogos de guerra, mas mais conhecido de todas é a simulação de batalhas num mapa que, desde o século XVII, é, simultaneamente, uma técnica de treino militar e uma forma bastante popular de entretenimento.

Os praticantes de jogos de guerra adoram regras complicadas, e H. G. Wells, que era um adepto entusiasta destes jogos, publicou,

há muitos anos, um livrinho chamado *Little Wars (Guerrazinhas),* que focava certos aspectos das simulações de batalhas sobre mapas e se tornou rapidamente a bíblia de todos os verdadeiros devotos deste passatempo.

Porém, durante a década de 60, foram comercializados livros de regras destinadas a jogos de guerra que atingiam níveis muito mais complexos. Estes livros, que se tornaram muito populares, tratavam, nos mais pequenos pormenores — uniformes, armas e logísticas —, os períodos considerados, que eram, essencialmente, a Antiguidade, a Idade Média, a era napoleónica, e os nossos dias.

MASMORRAS E DRAGÕES

Dentro das categorias acima consideradas havia muitas subcategorias, entre as quais se encontrava a chamada fantasia medieval. Dave Arneson, pertencente à Castle and Crusade Society, dos Estados Unidos da América, começou uma vasta campanha e expandiu as regras originais desta modalidade de forma a oferecer aos jogadores um cenário completo. Com a ajuda de Gary Gygax, estas regras converteram-se num dos jogos com maior sucesso no nosso século: *Dungeons & Dragons* (Masmorras e dragões), publicado pela Tactical Studies Rules. *D&D* apresenta ao jogador um sistema de jogo altamente estilizado, no qual nada é deixado ao acaso, sendo tudo jogado com referência a tabelas. Dado o sucesso de *Dungeons & Dragons,* era natural o aparecimento de muitas imitações, tendo algumas delas tido mais impacte que outras. De entre estes jogos, o mais duradouramente popular foi, provavelmente, *Tunnels & Trolls* (Túneis e Gigantes)[1], de Ken Andre, que simplificava bastante as regras de *D&D* mas se baseava em muitos dos aspectos que se tinham tornado tão populares no jogo anterior.

Tal como em *D&D,* um mestre construtor de masmorras contrói um conjunto de subterrâneos (ou qualquer outro tipo de cená-

[1] *Troll:* figura da mitologia nórdica, posteriormente utilizada por vários escritores de livros de aventuras fantásticas, entre os quais se destaca J. R. R. Tolkien. Não tem equivalente exacto em português. *(N. da T.)*

rio que se deseje), no qual é colocado o grupo de jogadores, que terão de se desembaraçar como puderem. A luta é moderada por jogo de dados, mas em *T&T* não existem dados de percentil. Os feitiços também são inerentes ao sistema *T&T,* progredindo em níveis conquistados à custa de duro esforço desde os inferiores, como *«knok-knok» (truz-truz),* que abre portas fechadas, e *«Take that you fiend»* (toma lá, malvado), que usa o quociente de inteligência como uma arma, passando por vários feitiços saborosamente baptizados como *«Zombie Zonc»* e *«Mutatum Mutandorum»,* até ao superior, *«Born Again»* (Nascido de novo), cujo nome fala por si.

Seja como for, o principal desenvolvimento de *T&T,* que tornou este jogo querido de muitos milhares de praticantes de jogos de fantasia em todo o mundo e o torna particularmente interessante para os utilizadores de computadores, é o seu sistema de «jogo a solo».

Este sistema baseia-se numa série de livros ilustrados, correspondendo cada um deles a uma aventura completa pronta a ser jogada, de acordo com as regras de *T&T,* por uma só pessoa. Estes livros, que se resumem a um conjunto de acções de escolha múltipla em que o texto desempenha o papel de «mestre construtor de masmorras», são uma bênção para o jogador que não pode encontrar-se com outros praticantes de jogos de aventuras.

O grande sucesso destes volumezinhos é um bom indicativo do número de pessoas que, não tendo possibilidade de jogar RPG (*role playing games* — jogos de representação de papel) completamente desenvolvidos, pode, graças a estes livros, encarar o computador como mediador, árbitro e mestre construtor de masmorras.

A relativa simplicidade e a clareza das regras do sistema *T&T* grangeou-lhe a preferência dos escritores de aventuras para computador baseadas na luta.

AVENTURAS

Nos meados da década de 70, dois indivíduos empreendedores chamados Willie Crowther e Don Woods inventaram, enquanto curvados sobre o seu enorme computador principal, um jogo a que chamaram *Colossal Cave* (Gruta Colossal) ou *Adventures* (Aventuras). É muito provável que fossem entusiastas de *D&D,* pois o cenário do seu jogo incluía um conjunto de grutas, povoadas por uma grande variedade de seres estranhos, onde se encontravam liberalmente espalhados tesouros de todos os tipos.

Tal como em *D&D,* o jogador vai-se deslocando lentamente por território desconhecido e recebendo informações sobre aquilo que o rodeia. Porém, neste caso, essas informações são-lhe fornecidas pelo computador: sendo um árbitro e mestre construtor de masmorras incansável, o computador é o meio ideal para a prática de jogos de fantasia.

No rasto de *Colossal Cave* apareceram outros jogos. Entre estes, o que alcançou maior sucesso foi, provavelmente, *Zork,* que é o precursor de muitos dos jogos de aventuras implementados para os microcomputadores pessoais nos últimos anos da década de 70. Durante vários anos, os aparelhos PET, Apple e Tandy estiveram bem servidos por estes jogos, enquanto o jogo original era passado em forma de disco entre os profissionais de computadores. No entanto, isto passava-se bastante no domínio do mercado negro, dado que as companhias possuidoras dos grandes computadores quer eram usados pelos seus empregados para jogar, ilicitamente, estes jogos, se preocupavam, como é óbvio, com este uso de dispendioso tempo de computador.

A atracção que estas aventuras moderadas por um computador têm para o entusiasta não profissional reside, evidentemente, no facto de elas poderem ser jogadas em qualquer altura, e até por um jogador solitário se tal for necessário (é, frequentemente, esclarecedor jogar com companheiros, dando cada um a sua contribuição própria). Apesar de o jogo poder, muitas vezes, levar semanas ou meses a completar, o estado do jogo num dado momento pode ser arquivado em disco ou fita, tornando-se viável, desta forma, retomar o jogo em qualquer altura.

A desvantagem, também óbvia, que este tipo de jogo apresenta para os entusiastas não profissionais é que, na sua forma original, as aventuras só podem ser passadas num computador principal, que custa uns bons milhares de contos — e não faz parte do mobiliário habitual duma sala de estar!

Com o aparecimento dos microcomputadores baratos, os programas de aventuras passaram a estar ao alcance de utilizadores não profissionais. A prática de jogos a solo tornou-se assim possível para os entusiastas que não são suficientemente afortunados para possuírem um computador principal IBM.

CAPÍTULO 2

Antecedentes

Na altura da elaboração deste livro, é anunciado um novo microcomputador de quinze em quinze dias. Alguns deles mostram ser aquilo a que se tem chamado «*vapourware*», ou seja, acabam por nunca se materializar. Apesar de isto ser um fenómeno perturbadoramente frequente no negócio dos computadores, ainda vão aparecendo microcomputadores no mercado, e até muito bons. A remessa inicial de *software* que acaba por chegar do fabricante de computadores inclui, invariavelmente, um programa de aventuras.

«ZX80» E «ZX81»

Há cerca de cinco ou seis anos, na pré-história da microcomputação, a cena era dominada por três aparelhos. Eram todos americanos, e muito caros — de resto, ainda são! Para todos eles tinham sido escritas implementações de aventuras e *Zork*. O *Apple I*, o *Commodore PET* e o *TRS-80* ainda se mantêm sob várias formas, mas o custo desses aparelhos nesses longínquos dias levava a que o clube dos praticantes de jogos de aventuras se mantivesse muito restrito.

Entrou então em cena Sir Clive Sinclair. Depois de ter, por si só, transformado o mercado dos relógios digitais com o *Black Watch* e o das calculadoras de bolso com a *Executive* (ainda que isto tenha sido devido ao seu baixo preço e aspecto atraente, e não

à sua longa duração), parece-nos, em retrospectiva, inevitável que ele se concentrasse em seguida no mercado dos computadores.

No entanto, o *ZX80* ainda era destinado a amadores entusiastas e o *software* comercial era praticamente inexistente. Só com o lançamento do *ZX81,* destinado ao mercado de largo consumo, é que as vendas de microcomputadores se desenvolveram e, com elas, a criação de *software* rendível. Isto, a propósito, parece ser a excepção que confirma a regra de que o *software* ajuda a vender o *hardware!*

Os programas de aventuras para o *ZX81* são abundantes: há aventuras de vários tipos, sendo algumas só texto e outras baseadas essencialmente em imagens, existindo ainda todos os tipos intermédios. Isto está de acordo com a tendência geral que se verifica nesta área de *software,* dado que alguns autores se mantêm fiéis à abordagem tradicional, enquanto outros preferem trilhar caminhos mais inovadores. Já falámos da génese das aventuras para computador com a aventura original de Crowther/Woods e *Zork,* mas nos meados da década de 70 também foram escritos nos Estados Unidos vários outros programas populares. À medida que os programadores se iam tornando mais eficientes nos seus novos brinquedos os aparelhos *ZX,* estes programas antigos foram sendo adaptados.

Muitos dos programas que foram convertidos tinham sido originalmente publicados na revista americana *Creative Computing,* de David Ahl. Entre os jogos mais populares contava-se *Hammurabi,* que hoje em dia é, com frequência, incorrectamente escrito *Hamurabi.* No original, o jogador tem de orientar o epónimo rei da Suméria na administração do reino. Dados tantos ares de terra arável, tantos alqueires de grão armazenado e tantos habitantes, o jogador tem de gerir todos os ponderáveis de forma que eles cheguem para um determinado número de anos. Este tipo de jogo tem sido muitas vezes adaptado, desde o seu primeiro aparecimento na revista *Creative Computing.* O nome geral dado a este tipo de jogo é «jogos de administração».

Muitos dos programas de aventuras que se encontram hoje em dia são, de facto, provenientes destes jogos de administração na sua cuidadosa gestão de vários ponderáveis. *Star Trek,* um jogo cuja primeira versão foi escrita nos finais da década de 60, no auge do entusiasmo pela série televisiva, é uma espécie de *Hammurabi,* dado que implica delicados malabarismos entre controlo de armas e

nível de munições, envolvendo cada operação de controlo ou de reparação de danos um determinado preço. Este tipo de jogo adapta-se idealmente ao computador, deixando a cargo do aparelho todos os pormenores de sondagem da galáxia e relatório de circunstâncias, e permitindo ao jogador dedicar-se a dar cabo dos Klingons. Porém, pode chegar o dia em que alguém escreva um programa de *Star Trek* cujo objectivo seja travar amizade com os extraterrestres...

Wumpus também foi escrito há muitos anos e sobreviveu até hoje — de facto até tem florescido. A versão original consistia na busca do ser mítico Wumpus através de um conjunto de quadrados. Utilizando pistas dadas pelo computador, o jogador acabava por restringir o número de escolhas possíveis e descobrir a localização final da fera. Em páginas futuras vamos encontrar descendentes de todos os programas de que temos vindo a falar: *Adventures, Zork, Hammurabi,* e *Wumpus.* Foram escritas muitas versões destes jogos para os primeiros aparelhos *Sinclair ZX80* e *81.* Estas versões estavam disponíveis comercialmente sob a forma de fita gravada ou de programas incluídos por extenso (listagens) num dos muitos livros escritos para estes aparelhos.

A Artic Computing, em particular, apresentou uma série extremamente interessante de aventuras imaginativamente baptizadas com o nome de *Adventures A, B* e *C* (bom, elas tinham os títulos alternativos de *Planet of Death* (O Planeta da Morte), *Inca Treasure* (O Tesouro Inca) e *Ship of Doom* (O Navio da Desgraça). Estas aventuras, que têm sobrevivido desde os primeiros dias do entusiasmo pelos *ZX,* adoptam o estilo de Crowther/Woods e são muito boas. O seu merecido êxito baseia-se em duas razões. A primeira é que os programas são, sem excepção, diabolicamente difíceis, mas, em última análise, compensadores. A segunda é que estão escritos directamente em código-máquina (isto numa altura em que a maioria dos autores do *software* ainda andava às voltas com o BASIC dos Sinclair), o que leva a que a resposta ao *imput* do jogador seja correspondentemente rápida.

Uma aventura da nossa preferência (e não pedimos desculpa por mencionar este facto!), *Catacombs* (Catacumbas) de J. K. Greye, é representativa da abordagem visual aos jogos de aventuras. Em *Catacombs,* a mecânica do jogo baseia-se muito em *Dungeons and Dragons* e em *Wumpus.* À medida que o jogador vai explorando um conjunto de subterrâneos do qual não possui nenhum mapa,

vai encontrando vários monstros. Quando o jogador atravessa o limiar de uma sala onde se encontra uma fera, o computador dá-lhe informações sobre esta. Desencadeia-se a luta e o combate prossegue até à derrota do jogador ou do monstro. As lutas vão-se tornando progressivamente mais difíceis à medida que vão sendo alcançados níveis mais elevados. Este jogo, que ainda se encontra à venda, é uma iniciação muito boa a este tipo de jogos de aventuras para o praticante em *ZX81*.

Outra abordagem, bastante moderna, às aventuras com imagens é *Fantastic Voyage* (Viagem Fantástica), da Foilkade. A acção, baseada no romance de ficção científica homónimo do jogo, desenrola-se — entre tantos lugares possíveis! — no corpo humano. Penso que este jogo não é exactamente um jogo de aventuras; no entanto, ele utiliza na prossecução do objectivo o mesmo tipo de mecânica de jogo que outros jogos muito mais tradicionais. Neste caso, o objectivo é a destruição de um coágulo no cérebro.

Estes são apenas dois dos muitos jogos de aventuras destinados aos primeiros aparelhos *Sinclair*. Porém, todos estes jogos tinham uma desvantagem: tendo sido escritos para os *ZX80* e *81*, não podiam deixar de ser a preto e branco. É claro que fazer esta crítica é bastante semelhante a dizer que não vale a pena ver nenhum filme a preto e branco ou que nenhum disco «mono» é boa música. Os programas clássicos permanecerão clássicos, mesmo que sejam silenciosos e monocromáticos, e a falta de som e de cor pode, ainda que raramente, ser uma virtude.

«SPECTRUM»

O aparecimento do *Spectrum* ocasionou um verdadeiro dilúvio de *software*, 95 % do qual está relacionado com jogos. Dessa percentagem, uma parte apreciável consiste em aventuras de vários tipos. A Artic Computing não modificou as aventuras que já estavam escritas para o *ZX81*, fazendo meras transcrições das aventuras completas, com texto negro sobre papel branco.

Mantendo a tradição, duas novas aventuras foram denominadas *E* e *F* (tendo como subtítulos *Golden Apples* — Maçãs Douradas — e *Ground Zero* — Terreno Zero —). Nestes programas, a

Artic Computing tirou partido do aspecto exterior do *Spectrum:* apresentou o texto em tinta furta-cores sobre papel preto. O resultado é muito colorido.

A aventura *E, Golden Apples* (tal como já tinha acontecido com as aventuras *A, B, C, e D*) é constituída apenas por texto, mas parece, às primeiras jogadas, bastante mais tradicional que os jogos anteriores. O programa foi escrito para uma memória maior, o que permite uma exploração dos locais mais pormenorizada. É uma pena que não pudessem ser comercializadas pela Artic versões das aventuras anteriores para 48K. Tal iniciativa teria ido certamente ao encontro de um amplo mercado. A aventura *F, Ground Zero,* escrita com a ajuda de *The Quill* (A Pena), da Gilsoft, é um encantador conto sobre a destruição nuclear. O jogador, sendo o protagonista, tem de sobreviver à conflagração final.

Outras editoras de *software* seguiram o exemplo da Artic, mantendo-se fiéis à abordagem tradicional. Entre elas destacam-se particularmente a Abersoft, a Foilkade e a Level 9. Enquanto tanto a Artic quanto a Foilkade (com *Adventure 200* — Aventura 200 —, cujo nome é uma referência ao número de locais incluídos no jogo) escreveram cenários próprios para as suas aventuras, a Abersoft e a Level 9 pegaram no clássico de Willie Crowther e Don Woods e traduziram-no para o *Spectrum;* no entanto, acrescentaram elementos da sua própria autoria.

Por exemplo, a Level 9, na sua *Colossal Adventure,* ampliou o original, acrescentando-lhe setenta locais. Duas continuações desta aventura, *Dungeon Adventure* (Aventura na Masmorra) e *Adventure Quest* (Em Busca da Aventura) desenvolvem-se a partir de locais existentes na aventura anterior. Estas três aventuras formam um interessante conjunto de jogos, extremamente difícil, mas sempre lógico e divertido.

Existem, evidentemente, outros programas que se apoiam em texto e que, seguindo o exemplo da Artic e da Foilkade, não se baseiam estritamente nas aventuras originais. São, geralmente, parentes afastados de *Wumpus* e de *Dungeons & Dragons* (e, às vezes — tal como sucede com *Sorcerer's Castle,* o Castelo do Feiticeiro, da Mikrogen —, de *Snakes and Ladders,* Serpentes e Escadas!). Entre estes programas incluem-se aventuras como *The Orb* (O Globo), da Quest Software, *Velnor's Lair* (O Covil de Velnor), da Neptune, e *Volcanic Dungeon* (Masmorra Vulcânica), da Carnell. Ainda que estes últimos programas possam incluir um limitado

apoio visual, como seja plantas topográficas e listas de armas, para pôr em relevo o texto, elas podem ainda ser consideradas como sendo aventuras em texto.

No entanto, o *Spectrum* exige que as suas possibilidades sonoras e cromáticas sejam utilizadas, e muitos autores tiraram de facto partido dessas características ao produzirem os seus programas de aventuras.

AVENTURAS EM TEXTO

Como vimos, os jogos de aventuras eram originalmente jogados nos grandes computadores principais. Estes aparelhos não desfrutavam do luxo de possuir um visor monitor. Em vez disso, o seu *output* era impresso (é esta a razão de ser de vários termos que se usam ainda hoje nos computadores pessoais, como sejam PRINT e TAB). As imagens estavam, pois, excluídas: em vez disso, o jogo era praticado utilizando resmas e resmas de papel de impressão.

Existem, hoje em dia, bandos de «aventureiros» reaccionários, defendendo cada um deles o seu tipo de aventura preferido. Os adeptos do texto acham que, sem imagens que provoquem na sua mente uma ideia preconcebida da cena, podem apreciar melhor a aventura, sendo a sua imaginação o único limite do divertimento; os entusiastas das imagens acham que a representação artística das cenas dá realce ao jogo. Pela nossa parte, adoramos aventuras de todos os géneros — gostamos de apreciar o trabalho de um bom artista, mas, simultaneamente, achamos que cada aventura em texto adquire uma personalidade própria à medida que a imaginação trabalha, dando vida às insípidas descrições (no entanto, o leitor vai ver que a Infocom e uma ou duas outras editoras utilizam descrições tão verbosas que deixam muito pouco lugar à imaginação!).

A típica aventura em texto começa com a descrição de um local. Na aventura original este local era o terreno circundando uma pequena cabana de madeira no meio da floresta. Esta descrição pode ser bastante lacónica: «*Você encontrado lado de fora de uma pequena cabana de madeira.*» Se a memória não for reduzida, os locais podem ser descritos de forma correspondentemente mais pormenorizada; no entanto, neste caso, a maioria dos autores pre-

fere introduzir no jogo mais locais. Se a aventura for fornecida em disco, cada descrição pode ser arquivada e lida pelo programa conforme for necessário, podendo assim as descrições ser muito mais pormenorizadas.

Depois da descrição, seja ela curta ou longa (ver a secção referente a aventuras da Infocom, no capítulo 3), o computador pergunta ao jogador que é que ele quer fazer. Há duas formas possíveis de as aventuras serem conduzidas. Depois da descrição do local, o computador pode dizer: «*Não vejo nada de interessante. Que é que faço agora?*» Neste caso, a aventura é jogada do ponto de vista do computador, que é dirigido pelo jogador. O computador torna-se, assim, numa espécie de marioneta do entusiasta de aventuras (ver, de novo, a secção dedicada à Infocom: como seria de esperar dos autores de *software* de aventuras mais originais da actualidade, este tipo de aventura na terceira pessoa é abordado de um modo completamente novo no seu programa *Suspended,* Suspenso).

No entanto, as aventuras na primeira pessoa são subtilmente diferentes. Neste caso, o computador diz: «*Você não vê nada. Que é que quer fazer agora?*» Isto torna tudo muito mais pessoal e imediato, e normalmente preferimos jogar este tipo de aventura.

As dificuldades que se encontram ao tentar descrever uma aventura em texto residem na tentativa de transmitir o clima essencial do jogo sem desvendar um enigma que pode andar a intrigar um leitor há muitas semanas, pois muito do divertimento de que se desfruta ao praticar jogos de aventuras em texto consiste em andar às voltas com um problema complicado até que, finalmente, se consegue resolvê-lo.

UMA AVENTURA TÍPICA

O que se vai seguir é uma situação hipotética, imaginada por mim, e que pode servir para exemplificar a maneira como se pode desenvolver uma situação típica. Na maior parte das aventuras, o objectivo subjacente é a apropriação de tesouros de vários tipos. Na minha opinião, qualquer objecto que possa ser apanhado e transportado pode ser incluído na categoria dos tesouros. No en-

tanto, alguns objectos, como chaves ou latas de óleo (muito útil para desenferrujar gonzos de portas!), têm uma utilidade prática na aventura, enquanto outros, tais como moedas de ouro, podem ser considerados tesouros «verdadeiros», e ter influência na pontuação final do jogador. Neste fragmento, o nosso jogador imaginário está a divertir-se bastante tentando apanhar todos estes tesouros!

Consideremos a situação seguinte:

Você está num largo corredor situado num conjunto de subterrâneos, onde chegou depois de ter coligido tesouros e outros objectos. Para saber o que está a transportar neste momento, tecle INVENTORY (inventário) — habitualmente basta teclar INV. O computador responde:

VOCE ESTA A TRANSPORTAR
O ELMO MAGICO
O ANEL DE OURO
O ESCUDO DE LATAO
O LIVRO DE QUADRINHOS
A OPALA DE FOGO
AS LUVAS CASTANHAS

VOCE ENCONTRA-SE NUM LONGO CORREDOR.
A OESTE VE UMA CAVERNA ESCURA.
E AGORA?

Bom, já agora podíamos entrar na caverna. Frequentemente, isto acaba por revelar-se perigoso, dado que o autor está provavelmente a pôr-nos à prova, e deve ter instalado todo o tipo de armadilhas ao longo do seu conjunto de subterrâneos. Muitas vezes a mensagem que se segue é:

VOCE ESCORREGOU NA ESCURIDAO E PARTIU
O CRANIO. QUER JOGAR OUTRA VEZ S/N?

No entanto, nós fomos antecipadamente avisados da falta de luz na caverna e não fomos informados da existência de nenhuma fonte de luz no corredor. Por isso, podemos razoavelmente presumir que, no interior da caverna, deve haver alguma espécie de luz. Tendo previamente salvaguardado (SAVE) a nossa posição (qualquer programa razoavelmente bom permite fazer isto antes de serem corridos quaisquer riscos, tornando assim mais fácil retomar rapidamente o jogo no ponto desejado, no caso de se ser morto), decidimos correr o risco de ir espreitar a caverna. Introduzimos:

W (Oeste)

E o computador responde:

AGORA VOCE ESTA NA CAVERNA ESCURA. UMA LUZ TENUE BRILHA NA ENTRADA POR DETRAS DE SI.
E MELHOR ARRANJAR UMA LUZ DEPRESSA.
E AGORA?

Tentemos o seguinte:

LE O LIVRO DE QUADRINHOS (para um leve[1] alívio, estão a perceber?)

É engenhoso, mas não resulta. O computador limita-se a repetir a sua mensagem.

ESTA ESCURO, AQUI. E MELHOR ARRANJAR UMA LUZ.

Isto não vai lá muito bem! Frequentemente, os programas de aventuras só nos deixam andar a tropeçar na escuridão durante um curto período ou um par de interacções com o computador, até chegarem à conclusão de que não temos a menor ideia do que vamos fazer a seguir. Nessa altura, mergulham-nos sem qualquer cerimónia na água gelada do ribeiro subterrâneo mais próximo ou fazem-nos chocar contra um rochedo saliente. Seja qual for o método escolhido, o programa leva o jogo a uma rápida conclusão.

Mas que é isto? Aparece outra coisa no visor.

VOCE SENTE AS SUAS PERNAS SEREM ENVOLVIDAS POR TENTACULOS PEGAJOSOS. E UM PEQUENO POLVO.

---

[1] Em inglês «leve» e «luz» correspondem à mesma palavra, *light*. Daí o trocadilho. *(N. da T.)*

O polvo é capaz de dar alguma ajuda neste caso. Por isso:

APANHAR O POLVO

Mas o computador responde:

BEH! NAO, MUITO OBRIGADO. E MUITO PEGAJOSO.
E AGORA?
POR AS LUVAS
JA ESTA. AS LUVAS CASTANHAS ESTAO POSTAS.
E AGORA?
APANHAR O POLVO
O POLVO JA ESTA APANHADO. MUITAS MAOS TORNAM O TRABALHO LEVE[1]*. UMA LUZ SUAVE EMANA DE UMA FONTE INVISIVEL. VOCE ESTA NUMA PEQUENA CAVERNA ONDE SE ENCONTRA UMA PEQUENA CAIXA DE MADEIRA. PARA NORTE VE UM RIBEIRO SUBTERRANEO DESLIZANDO ATRAVES DE UMA PEQUENA FENDA NA ROCHA. PARA LA DA ROCHA APENAS CONSEGUE VISLUMBRAR QUALQUER COISA QUE BRILHA NA ESCURIDAO.*
E AGORA?

Ah! Mais tesouros! Temos de os ir apanhar. Impantes de justificável orgulho pelo nosso êxito na utilização do polvo, atacamos com alegria o problema seguinte. Como é que vamos alcançar esse ouro que está na próxima caverna?

Podemos tentar o caminho directo, introduzindo:

N (norte)
NAO PODE IR POR AI.
E AGORA?

É uma bela desilusão. O computador, diabos o levem!, não parece estar nada impressionado com as nossas brilhantes deduções anteriores, e já está a colocar mais dificuldades no nosso caminho.

---

[1] *Trocadilho envolvendo a palavra light* que tanto quer dizer luz como leve. Assim, *make light work* tanto pode querer dizer «tornar o trabalho leve» quanto «fazer a luz funcionar». *(N. da T.)*

Tem de haver uma maneira de transpor a fenda. Talvez estejamos a transportar demasiados objectos e não possamos esgueirar-nos através dela.

POUSAR TODOS OS OBJECTOS
ESTA BEM.
E AGORA?
N
NAO PODE IR POR AI.
E AGORA?

Que tal mergulhar no ribeiro? Talvez consigamos nadar ou flutuar através da fenda.

MERGULHAR NO RIBEIRO
DEVE ESTAR A BRINCAR, EU NAO POSSO RESPIRAR DEBAIXO DE AGUA.
E AGORA?

Bolas, isto está a ficar parvo! Haverá na caverna alguma coisa que nos possa ajudar? No chão está o pequeno monte de tesouros que pousámos há poucos momentos, mas nenhum é útil nesta altura. É verdade! A caixa! Como foi que pudemos esquecer-nos dela?

EXAMINAR A CAIXA
VOCE VE UMA PEQUENA CAIXA DE MADEIRA.
A CAIXA NAO TEM NADA DE ESPECIAL.

Como os computadores podem ser pretensiosos!

ABRIR A CAIXA
ESTA BEM. A CAIXA ESTA ABERTA. NO INTERIOR VE-SE UM DISPOSITIVO DE RESPIRAÇAO SUBAQUATICA.

Dispositivo de respiração subaquática? Ribeiro subterrâneo?

APANHAR O DISPOSITIVO DE RESPIRAÇÃO SUBAQUATICA
ESTA BEM. JA TEM DISPOSITIVO DE RESPIRAÇAO SUBAQUATICA.
E AGORA?

Bom, e agora? Temos o dispositivo de respiração subaquática, e há uma hipótese razoável de que agora consigamos resolver o problema do ribeiro.

Porém, queremos levar connosco todos os nossos belos tesouros, e a água tem o hábito aborrecido de correr num só sentido. Podem apostar o vosso último anel de ouro em como, um vez no ribeiro, não conseguimos voltar atrás. Por isso, temos de enfrentar a longa tarefa de voltar a apanhar todos os objectos que queremos levar connosco. No entanto, o computador tem outras ideias.

VOCE LEVA O DISPOSITIVO DE RESPIRAÇAO SUBAQUATICA.
SO PODE LEVAR MAIS UM OBJECTO.

É óbvio que o dispositivo de respiração subaquática deve ser demasiado volumoso ou pesado. Vamos só recapitular os objectos que se encontram na caverna neste momento.

OLHAR
VOCE: VE
O ELMO MAGICO
O ANEL DE OURO
O ESCUDO DE LATAO
O LIVRO DE QUADRINHOS
A OPALA DE FOGO
O POLVO QUE O ESTA A OBSERVAR.
VOCE TRANSPORTA:
O DISPOSITIVO DE RESPIRAÇAO SUBAQUATICA
AS LUVAS QUE TEM CALÇADAS
ALEM DISSO HA UMA PEQUENA CAIXA ABERTA.

É muita tralha para transportar. Espera aí! A caixa pode ser útil. Foi-nos dito que é pequena, mas talvez lá caiba tudo o que queremos levar, e pode ser que ela flutue!

Agora podemos tentar:

PEGAR NO ANEL
ESTA BEM. JA TEM O ANEL
E AGORA?
POR O ANEL NA CAIXA

ESTA BEM.
E AGORA?
OLHAR

Como resposta obtemos a mesma lista que anteriormente, mas com uma diferença no fim.

E UMA PEQUENA CAIXA ABERTA QUE CONTEM O ANEL DE OURO.

Resultou! Agora temos de repetir o processo com o resto dos tesouros. Depois de alguns momentos passados a dactilografar, outra instrução para OLHAR informa-nos de que a caixa contém todos os nossos tesouros tão arduamente conquistados. De momento ficamos com o polvo. Agora vamos tentar:

ATIRAR A CAIXA
PARA ONDE?
PARA O RIBEIRO
ESTA BEM. A CAIXA E TRANSPORTADA RAPIDAMENTE ATRAVES DA FENDA NA ROCHA E PERDE-SE DE VISTA.

Agora vamos fazer o mesmo com o polvo. O computador diz-nos que ele também desapareceu da nossa vista. Agora escrevemos:

SALTAR PARA O RIBEIRO

E o computador responde:

DEVE ESTAR A BRINCAR. EU NAO POSSO RESPIRAR DEBAIXO DE AGUA.

Isto é tão aborrecido! Vamos ter de executar todos os passos para benefício do computador.

POR O DISPOSITIVO DE RESPIRAÇÃO SUBAQUATICA
ESTA BEM. O DISPOSITIVO DE RESPIRAÇÃO SUBAQUATICA JA ESTA POSTO. VOCE ESTA A SUFOCAR. FAÇA QUALQUER COISA, DEPRESSA!

Bolas! Devemos ter-nos esquecido de alguma coisa!

LIGAR A VALVULA DE AR.
ESTA BEM. A VALVULA DE AR JA ESTA LIGADA.
E AGORA?
SALTAR PARA O RIBEIRO
ESTA BEM. VOCE E TRANSPORTADO PELA RAPIDA CORRENTE DO RIBEIRO ATRAVES DA FENDA DO ROCHEDO. E ENTRA NUMA CAVERNA. O POLVO JA LA ESTA. MUITAS MAOS TORNARAM O TRABALHO LEVE. ACIMA DE SI ENCONTRA-SE A MARGEM DO RIBEIRO.

Suspiramos de alívio. Agora temos de subir para a margem do ribeiro.

SUBIR
ESTA BEM.
VOCE ENCONTRA-SE NUMA GRANDE CAVERNA.
VE UMA PEQUENA CAIXA QUE ESTA ABERTA
E CONTEM MUITOS TESOUROS.

(Mas esperem um momento antes de irem buscar os tesouros!).

VOCE TAMBEM VE:
UM DRAGAO ADORMECIDO. AS ESCAMAS DELE REFULGEM SOB A LUZ.
NEM TUDO O QUE BRILHA E OURO! ELE ESTA A COMEÇAR A MEXER-SE.
E AGORA?

Mas que pergunta! Bom! E agora!

Eis um exemplo dos programas típicos que podem encontrar-se numa aventura tradicional (não digo de qual, não vá dar-se o caso de estar a jogá-la nesta altura): uma série de problemas interligados que finalmente levam a um resultado. Esse resultado pode ser, como no nosso exemplo, uma quantidade de sarilhos, mas claro que existe sempre a possibilidade de ser conduzido a um tesouro.

Todavia, alcançar o tesouro não é o fim da história, uma vez que é necessário regressar a um determinado local para depositar o tesouro e coligir os pontos.

LEVEL 9

Vamos agora ver algumas aventuras da Level 9, em particular a *Dungeon Adverture* (Aventura na Masmorra).

Qualquer leitor que esteja presentemente a jogar esta aventura deve saltar o resto desta secção. Aqueles que estão a pensar em comprar este programa não devem desesperar: não vai ser desvendado nenhum facto crucial!

A *Dungeon Adventure* faz parte de uma excelente série de aventuras escrita pela Level 9, de High Wycombe, que começa com *Colossal Adventure,* muito inspirada no programa original de Crowther Woods, *Colossal Cave,* mas acrescenta cerca de setenta locais ao jogo final. O primeiro programa desta série, chamada *The Middle Earth Adventures* (As aventuras da Terra do Meio) é *Adventure Quest* (em Busca da Aventura). Todas as *Middle Earth Adventures* se inter-relacionam, e, em conjunto, formam um grande mundo fantástico que pode ser explorado e apreciado.

Snow Ball (Bola de Neve) é a primeira aventura de outra série da Level 9, chamada *Silicon Dream* (Sonho Sintético). Esta aventura desenvolve-se numa enorme nave espacial, com cinco milhas de comprimento. Esta aventura dispõe de mais de sete mil locais (pelo menos a Level 9 assim o afirma), e de rouxinóis comedores de homens, crescendo descontroladamente e patrulhando os corredores da nave espacial. O título faz provavelmente referência às possibilidades de sobrevivência do jogador! A série *Silicon Dream* promete ser tão boa quanto a *Middle Earth.*

Outro título da Level 9 é *Lords of Time* (Os Senhores do Tempo), a primeira aventura de uma outra série. Esta aventura escrita por Sue Gazzard é uma maravilhosa correria por entre nove sectores do tempo, durante a qual o jogador é testemunha de uma fascinante panóplia de cenários.

A cena de abertura de *Dungeon Adventure* (bem como os quebra-cabeças que lhe estão associados) é um exemplo muito puro do programa de aventuras clássico, e vai servir para a elaboração de algumas regras básicas respeitantes à prática de aventuras em geral.

A pequena brochura que acompanha a cassete de *Dungeon Adventure* é típica da qualidade da produção da Level 9. Uma página de exposição da cena, com um breve apontamento histórico

sobre o cerco de Minas Tirith, a morte do Senhor dos Demónios e a decisão do protagonista de se retirar rapidamente e ir procurar o tesouro do Senhor na Torre Negra, prepara o jogador para o começo do jogo. O resto da brochura esquematiza as várias instruções que podem ser dadas ao computador pelo jogador e dá várias sugestões ao praticante de aventuras. Para finalizar, encontramos uma particularidade única: um sobrescrito por intermédio do qual o jogador pode, se se encontrar lamentavelmente encalhado, pedir à Level 9 que lhe forneça uma pista.

No começo da aventura você, o herói, acorda, com frio e desarmado, sobre a margem lamacenta de um rio um grande caixote aberto de um dos lados. Encontra-se também perto de si um pedaço de madeira trazida pelas águas. Vê-se uma ponte de pedra que atravessa o rio, erguendo-se entre os penhascos de granito acima de si e as terras baixas da margem oposta. É claro que o jogador se pode dirigir arrebatadamente para a ponte, ansioso por começar a explorar as cavernas que, secretamente, sabe existirem para esse lado. E assim chegamos à Primeira Regra de Ouro da prática de jogos de aventuras:

Observar cada local o mais pormenorizadamente possível.

No começo das aventuras tradicionais, de que estamos agora a tratar, o jogador encontra-se quase invariavelmente sozinho e indefeso no exterior de um edifício de qualquer tipo.

No caso presente, há cerca de cento e setenta locais a explorar no conjunto de cavernas que o jogador vai encontrar do lado de lá da ponte. No entanto, são deprimentemente escassas as probabilidades de que ele sobreviva e venha a ver mais que uma meia dúzia deles.

Ainda muito perto do princípio, o jogador vai sentir necessidade de certos objectos que só podem ser obtidos nos locais exteriores às cavernas, ou seja, nos locais que ele encontra no início, do lado de cá da ponte.

O jogador vai descobrir tudo isto depois de ter feito algumas tentativas para se desembaraçar no conjunto de cavernas. Agora podemos começar a procurar esses objectos vitais. Sabemos que não podemos encontrar nada útil (por enquanto) indo para Norte, para lá da ponte, por isso tentemos dirigir-nos para Sul.

Porém, antes de mais nada, não devemos esquecer a nossa primeira regra de ouro da prática de aventuras. *Observemos o que nos rodeia o mais pormenorizadamente possível.* Se bem estão lembrados, encontramo-nos numa margem lamacenta, e podemos ver alguma madeira trazida pelas águas e um grande caixote. Vamos apanhar a madeira. Uma vez que programas diferentes reconhecem instruções diferentes para esta operação, podemos tentar PICK, TAKE ou GET (formas aproximadamente sinónimas significando «apanhar»). Em *Dungeon Adventure* é aceite TAKE. Assim, somos agora os orgulhosos proprietários de um pedaço de madeira.

De momento, a madeira não parece ter grande utilidade. Por isso, focamos a nossa atenção noutra coisa. Como se devem lembrar, o caixote estava aberto de um dos lados. Não lhes parece que era uma boa ideia ir lá dentro? Não era, não, pelo menos por enquanto! Os escritores de programas de aventuras pertencem ao género sádico, e haviam de adorar que o jogador, confiante e inocentemente, caísse numa armadilha — quem sabe se igualzinha a esta. Por isso, vamos tentar usar outra palavra-chave — EXAMINE N (examinar N, sendo N o objecto a ser examinado) — que se encontra muitas vezes em aventuras tradicionais como esta.

Assim, introduzamos EXAMINE CASE (examinar o caixote) e vejamos o que o computador tem para nos dizer. O caixote é encantado! Além disso, há espaço para rastejarmos lá para dentro. Agora sabemos que não vamos ser devorados nem espezinhados por nenhum monstro que venha a passar, e, de momento, continuamos a salvo.

No entanto, deixemos o caixote aqui. É claro que dentro dele há segredos a serem desvendados, dos quais o menor não é o facto de o caixote ser o local onde o jogador deve depositar os seus tesouros para obter pontos e ganhar o jogo. Mas é divertido tentar resolver as adivinhas que conduzem a esta informação, e não queremos estragar o vosso divertimento num estado tão pouco adiantado do jogo.

Portanto, vamos partir do princípio de que explorámos convenientemente o caixote e já estamos de novo cá fora sobre a margem lamacenta, com o nosso pedaço de madeira. Depois de várias tentativas dolorosas acabámos por chegar à conclusão de que tentar atravessar as escuras cavernas a Norte da ponte é uma jogada difícil. Visto isso, tentemos ir para Sul da ponte. Estamos a jogar esta aventura à medida que escrevemos, e isto é uma viagem de descoberta para todos nós!

Agora encontramo-nos numa estrada com a direcção Leste-Oeste, a Sul do rio, que é atravessado pela ponte. Uma gigantesca cabeça de orca está grosseiramente esculpida no penhasco a Norte do rio, e é a sua língua que forma a ponte. No cimo do penhasco ergue-se uma torre arruinada. As saídas são Leste, Oeste e Sul. Esta é a fórmula-padrão para aventuras em texto: entra-se num local do qual é fornecida uma breve descrição juntamente com uma lista das saídas possíveis. Na realidade, nas aventuras da Level 9 as descrições são bastante pormenorizadas.

Geralmente é fornecida também outra lista: a dos objectos e/ou monstros que se encontram à vista, e das coisas que o jogador está a transportar. No entanto, não há nada à vista neste local, por isso temos de decidir qual é a direcção que vamos tomar. Neste ponto não temos quaisquer pistas; atiremos uma moeda ao ar e tomemos a estrada Leste. O local seguinte continua a ser a estrada Leste-Oeste, mas encontramo-nos mais afastados do ponto de partida e podemos ver para Sul uma planície verdejante que se estende até onde a vista alcança.

Esta é uma dessas frases que o jogador acaba por considerar com uma certa apreensão. Geralmente, ela significa que, se entrar no local a que ela se refere, o jogador fica perdido para sempre — ou, pelo menos, até desistir, completamente desesperado!

Assim, vamos ignorar, de momento, a planície verdejante. O computador informa-nos de que também podemos ver uma fieira de pedras chatas que emergem das águas, conduzindo a uma pequena ilha. Uma rapariga de caracóis esvoaçantes está sentada na ilha. É encantador! Teclamos GO NORTH (ir para Norte). O computador informa-nos de que, agora, nos encontramos sobre as pedras, e de que a rapariga ainda está sentada na ilha. Introduzimos de novo GO NORTH. Valha-nos Deus! A rapariga, que é uma sereia, canta a sua canção; nós debatemo-nos nas traiçoeiras águas do rio. O computador informa-nos de que arranjámos maneira de morrer.

Agora, o programa faz-nos saltar de novo para a margem lamacenta e temos de refazer o nosso caminho até à estrada Leste-Oeste, mesmo a Sul da ponte. Desta vez, vamos explorar para Leste. Introduzimos GO EAST (ir para Leste) e, adequadamente, o computador diz-nos que nos encontramos na estrada, mais para Leste, a Norte de um monte íngreme e despido de vegetação.

O monte parece digno de ser investigado. Dirigindo-nos para

Sul, chegamos ao sopé do íngreme monte, que se eleva até às nuvens. Ouve-se um ribombar proveniente do alto do monte. Podemos subir (UP) até lá; por isso, vamos! Agora estamos num círculo de monólitos distorcidos — figuras grotescas esculpidas pelas chuvas ácidas. Acima de nós faíscam relâmpagos, revelando este local assombrado em todo o seu horror, enquanto os trovões ecoam em cada rochedo. Tem um ar muito acolhedor, não é verdade? Se dermos ao computador instruções para olhar (LOOK) em volta, aparecerá subitamente uma hoste de ferozes labaredas. Estas chamas são Rakshasa! O seu chefe flutua em relação a nós e desafia-nos para um jogo: se o jogador for o vencedor da partida de dados verá de alguma forma aumentada a sua força espiritual; mas claro que também pode perder...

Temos de jogar. Bem vistas as coisas, parece um bom negócio, não acham? Infelizmente, precisamos de descobrir uma forma de viciar os dados (estes Rakshasas são muito espertalhões), mas ainda não encontrámos nenhuma, apesar de eu afirmar que ela deve poder ser descoberta em qualquer lado, nesta sequência de abertura.

Como já deviam estar à espera, perdemos este jogo, e assim encontramo-nos de novo na margem lamacenta. Seja como for, vamos aprendendo.

Vamos voltar ao ponto da estrada a Norte do monte íngreme onde acabámos de encontrar os Rakshasas. Não voltaremos lá até termos encontrado a forma de ganhar o jogo de dados. Assim, continuamos para Oeste, para ver o que nos espera.

Agora vemos um vasto campo de papoulas que se estende até onde a vista alcança (onde é que eu já ouvi isto?). Temos de andar com muito cuidado aqui. Um perfume sonolento voga suavemente na brisa. No chão, perto de nós, podemos ver uma semente de papoula seca. Vamos apanhá-la, teclando TAKE SEED (apanhar a semente). E chegamos aqui à nossa Segunda Regra de Ouro da prática de jogos de aventuras:

Tudo tem uma finalidade.

O programa de aventuras típico está escrito de forma tão condensada que o programador não pode dar-se ao luxo de introduzir demasiados elementos inúteis. No entanto, como acontece com todas as regras, esta também pode ter uma excepção: o jogador pode encontrar o esporádico chamariz que o fará correr para todo o lado

em busca de uma forma de o utilizar. Além disso, é claro, encontra-se por vezes um autor verdadeiramente malévolo que se delicia colocando nas mãos de um incauto aventureiro uma bomba disfarçada.

Assim, apesar de podermos utilizar esta regra de ouro, temos de lhe acrescentar uma adenda:

Ser sempre extremamente cauteloso.

Vamos agora voltar para trás, pois temos um pressentimento estranho acerca desse campo de papoulas.

Podemos voltar para trás pela estrada, passando pelo monte íngreme e pela ponte, e dirigimo-nos de novo até às pedras que levam à ilha. Talvez seja possível utilizar a semente de papoula para combater a sereia (ainda não sabemos se resulta, mas vale a pena tentar!). Junto às pedras introduzimos NORTH, o que nos deixa sobre elas. Da última vez que aqui estivemos, fomos mergulhados na água sem cerimónia nenhuma, ao tentarmos passar pela sereia para podermos continuar a avançar para Norte. Desta vez vamos ser mais cautelosos.

Podíamos experimentar agora a semente de papoula. Vamos teclar THROW SEEDS (atirar as sementes) e ver o que acontece — estamos a concentrar aqui um grande desejo de conseguir passar e um grande esforço para encontrar a solução correcta. Ao cair, as sementes eclodem em ruidosas explosões. Será que assustam a sereia?

Para o descobrir, temos de dar uma instrução ao computador. Teclamos LOOK (olhar), e o computador diz-nos amavelmente quais são os objectos que se encontram no local. É evidente que, muitas vezes, a resposta que se obtém é *«não vejo nada de especial»;* no entanto, esta é uma pergunta que, tal como EXAMINE, vale a pena fazer. LOOK não deve ser confundido com REDESCRIBE (voltar a descrever) — abreviatura REDES ou R —, que dá ao computador instruções para voltar a descrever o local em que o jogador se encontra nessa altura.

Mas voltemos ao nosso actual problema. Tendo tentado LOOK, descobrimos que a sereia ainda está sentada ao pé do rio! Vamos introduzir outra palavra útil, LISTEN (ouvir). Os seres perigosos têm o hábito desagradável de se esconder por detrás de árvores ou rochedos, fora do campo de visão do computador, mas

não podem impedir-se de fazer barulhos deslizantes e furtivos, que os traem!

Ah! Estamos a obter alguns resultados. O computador diz-nos que ficámos temporariamente ensurdecidos e não podemos ouvir nada. A sereia ainda está, com certeza, a cantar como qualquer sereia que se preza; porém nós não a ouvimos, por isso talvez seja seguro passar por ela sub-repticiamente. Mas ela apanhou-nos outra vez! O efeito de surdez temporária deve ser muito curto.

E cá estamos nós de novo na margem lamacenta! Temos de voltar rapidamente ao local onde essa malvada dessa sereia se encontra, sem nos esquecermos de ir previamente buscar a semente de papoula.

Outra palavra de que vale a pena lembrarmo-nos, porque nos vai ajudar no futuro, é SAVE (salvaguardar). A maioria dos bons programas de aventuras permite a utilização dessa instrução que significa que é possível arquivar o estado actual do jogo numa fita ou disco à parte. Caso seja possível, deve fazer-se isto antes de dar qualquer passo arriscado. Devíamos ter sido mais prudentes e ter salvaguardado a nossa posição antes de tentarmos passar pela sereia: bastar-nos-ia agora teclar RESTORE (reestabelecer) e, uma vez retomada a nossa anterior posição, fazer uma nova tentativa de atravessar as pedras. Como não fomos prudentes, vamos ter de nos fartar de escrever no teclado para voltarmos, de semente na mão, ao local onde a sereia se encontra. Assim, façamos disto a nossa Terceira Regra de Ouro da prática de aventuras:

Em caso de dúvida, o jogador deve sempre salvaguardar (SAVE) a sua posição.

Lá estamos nós de novo junto da sereia. Deixámos cair a semente que explodiu devidamente. Salvaguardemos (SAVE) a nossa posição, não vá dar-se o caso de sermos de novo desfeiteados, e vamos a isso: passemos por ela o mais depressa possível, pois já sabemos qual é o efeito das explosões. Ah! Resultou! A sereia, ao ver que a sua canção não tem efeito sobre nós, entra em pânico e foge. De facto, neste ponto, o programa autoriza o jogador a introduzir até quatro instruções antes que a surdez desapareça.

Eis outro estratagema que é utilizado nos melhores programas de aventuras: ao abordar um problema de uma determinada manei-

ra pode obter-se um resultado completamente diferente daquele que é obtido noutra tentativa.

Agora encontramo-nos na extremidade de uma pequena ilha, cujo ponto mais afastado está ocupado por um salgueiro de aspecto malévolo com seis ramos elásticos. Junto dele, no chão, está um espelho de prata. Nós podíamos apanhar (TAKE) o espelho, mas, se o tentássemos e passássemos pela árvore, seríamos mortos. Qual é a razão de ser dos *seis* ramos? Deve haver aí uma pista!

Deixemos agora a *Dungeon Adventure*. Este jogo é extremamente rico. Falámos apenas de um pequeno número de locais e não mencionámos o processo de ressurreição de que o jogador vai precisar para jogar mesmo a sério, nem vários pormenores importantes que se encontram no cenário de abertura.

De facto, esta sequência toda, que se compõe de cerca de trinta locais, não passa de uma introdução à aventura principal, que tem lugar no conjunto de cavernas de que falámos. Ao começar o programa de aventuras original, o jogador encontrava-se no exterior de um pequeno edifício na floresta. No interior deste edifício estavam vários objectos que eram necessários durante a aventura, tais como chaves, comida e água. Além disso, todos os tesouros tinham de ser trazidos para o edifício. A larga rede de locais que se encontra no início da *Dungeon Adventure* tem a mesma finalidade sob uma forma muito mais complexa: o princípio da aventura constitui realmente por si só uma intrigante miniaventura.

Após as nossas primeiras e desajeitadas tentativas na *Dungeon Adventure,* os leitores devem ter ficado com uma ideia do sentimento que se experimenta ao jogar uma aventura em texto (e deve sentir-se motivado para jogar as aventuras da Level 9).

Entre o restante *software* britânico, pouco há que possa competir com a mistura de humor, complexidade, nível literário, apresentação de quebra-cabeças e absoluta temeridade que é particular à Level 9. Com estes programas, o jogador encontra-se alternadamente tremente de frustração ou perdido de riso incontrolável. Imagine, por exemplo, o momento de uma das aventuras desta editora em que o jogador é confrontado com uma planta pequenina que murmura, numa voz mansa e lamentosa: «água... água». Apiedado da pobre criaturinha, o jogador rega-a e vê depois, impotente, a planta erguer-se a dois metros e meio de altura e ameaçá-lo, exigindo: «ÁGUA, ÁGUA!!!»

CHANNEL 8

Uma firma britânica de *software* que pode desafiar a Level 9 é a Channel 8 Software, de Preston. A Channel 8 tem cerca de dez aventuras para os microcomputadores *Atari, Spectrum, BBC, Commodore 64, Dragon 32* e ainda outros aparelhos. O *pedigree* do catálogo é longo e honroso, pois ele foi inicialmente escrito, ainda em 1980, por Brian Howarth para os aparelhos *TRS-80* e *Nascom,* tendo sido comercializado através da Molimerx. Segue-se uma lista destas aventuras, que têm nomes bem intrigantes:

The Golden Baton (O Bastão de Ouro): Desde que o bastão de ouro foi roubado do reino do rei Ferrenuil, têm sido enviados valentes soldados e experimentados cavaleiros em busca dele para as mais longínquas paragens. Nenhum jamais voltou... Agora é a vez do jogador!

The Time Machine (A Máquina do Tempo): O jogador, repórter da *Tulkingham & Dunsby Gazette,* está sozinho, envolvido pelo nevoeiro que cobre a charneca. Vai começar a investigar os estranhos acontecimentos que se têm verificado à volta da velha casa...

Arrow of Death (A Flecha da Morte): Finalmente, o jogador recuperou o bastão de ouro e restituiu-o ao rei Ferrenuil. No entanto, o bastão de ouro tornou-se recentemente um símbolo do ódio e do mal. Possuído por um pressentimento de perigo iminente, o jogador dirige-se para o palácio. Como irá um mero mortal, como ele, conseguir opor-se ao poder maléfico que ameaça o futuro da sua terra?

Arrow of Death Pt 2 (A Flecha da Morte, 2.ª parte): Tendo completado com sucesso a primeira parte, o jogador vai precisar da Flecha para derrotar Xerdon, *o Mau!*

Escape from Pulsar 7 (Fuga da Pulsar 7): O jogador encontra-se sozinho, a bordo da Pulsar 7, e é perseguido por uma criatura selvagem que já matou e devorou o resto da tripulação.

Circus (Circo): O jogador, que se encontra sem gasolina numa escura azinhaga campesina, dá com um circo no meio de um campo. Os roucos sons de risos e de alegria extinguem-se mal ele se aproxima. Está a começar a escurecer, e a noite vai ser muito, muito longa!

Feasibility Experiment (Experiência Exequível): O jogador acorda de um sono agitado e descobre que está a ser utilizado como sujeito experimental por estranhos extraterrestres, seres incorpóreos, vindos de um mundo artificial muito para além dos limites da galáxia (de onde é que havia de ser?). O seu objectivo é estudar a espécie humana, na mira de encontrarem um gene do heroísmo, a ser incluído no seu património genético. O heroísmo do jogador vai ser testado numa arena.

The Wizard Akryz (O Feiticeiro Akryz): O malvado feiticeiro está farto do metediço mortal que tem deitado a perder todas as suas tentativas de se apoderar do bastão de ouro. Agora, o feiticeiro imaginou o mais terrível dos planos. O jogador é esse mortal, e esta aventura poderia chamar-se *Golden Baton Pt 3* (O Bastão de Ouro, terceira parte).

Perseus and Andromeda (Perseu e Andrómeda): o jogador é Perseu, e a sua tarefa é ir buscar a cabeça da Medusa, a Górgona, cujo olhar transforma em pedra homens adultos.

Ten Little Indians (Dez Indiozinhos): A imprensa tem dado muita publicidade ao tesouro do major Johnston-Smythe, que está escondido algures na sua tenebrosa mansão. O jogador não consegue resistir ao desafio de tentar descobri-lo e, em consequência disso, encontra-se num comboio, atravessando o campo em direcção ao seu destino.

Waxworks (Figuras de Cera): Esta é a mais recente das misteriosas aventuras. Tendo passado o dia numa praia quente, o jogador decide descansar na fresca atmosfera do museu das figuras de cera. Mas, como todos sabem, as figuras de cera não são exactamente o que parecem...

Quando isto for lido, já terão sido lançadas mais uma ou duas aventuras — tarde de mais, porém, para que eu possa ter visto algum exemplar delas. Os seus nomes são de novo intrigantes: *Midwinter* (No Meio do Inverno) e *After the Fire* (Depois do Fogo), uma aventura pós-nuclear.

As versões para o *Commodore* e o *Spectrum* contêm imagens de alta resolução, que podem ser desconectadas quando o jogador quiser. Quando estão desconectadas, aparece no visor, em seu lugar, uma lista de informações relevantes, que inclui uma lista de objectos úteis. No essencial, os programas seguem o padrão habitual do *input* de duas palavras e, assim, não oferecem surpresas ao praticante de jogos de aventuras iniciado através de Scott Adams. Howarth até segue as pisadas de Adams ao dedicar todas as aventuras (é verdade que algumas das dedicatórias são obscuras, mas isso é próprio das dedicatórias).

Como vimos, os enredos e os cenários são pouco habituais e constituem uma agradável mudança em relação à pancadaria contra gigantes que se encontra em muitas outras aventuras. É extremamente difícil dar com a solução de uma grande parte dos quebra-cabeças (claro que, quando são resolvidos, acabou-se, pois só há uma maneira de chegar ao fim destas aventuras). As aventuras em si e os atraentes envólucros em que são apresentadas, que fazem boa figura em qualquer estante, criam uma série interessante que compensa a atenção que lhe é prestada.

AS BASES DAS AVENTURAS

Nas próximas páginas vamos tratar de várias áreas gerais de jogos de aventuras, para dar ao leitor uma ideia dos muitos tipos de problemas que ele irá encontrar.

AS REGRAS DE OURO

Mencionámos três Regras de Ouro na secção sobre *Dungeon Adventure,* mas é evidente que o leitor vai conseguir encontrar outras.

VOCABULÁRIO

Tal como existe uma discussão sobre as vantagens e desvantagens do texto e das imagens, existe uma discussão sobre o vocabulário: revelar ou não revelar, eis a questão. Algumas aventuras desvendam tudo na documentação, e, neste caso, todas as palavras aceites pelo computador vêm escritas preto no branco, não havendo confusões quando se joga. No entanto, a maior parte das aventuras acha necessário acrescentar aos quebra-cabeças inerentes ao jogo a «Grande Caçada às Palavras»: neste caso, apenas uma pequena parte das palavras aceites é divulgada na documentação. Às vezes não chega sequer a ser divulgada qualquer palavra. No primeiro caso, é dado ao jogador o estritamente indispensável para que possa começar; no segundo, parte-se do princípio de que o jogador é um «aventureiro» consumado, e, portanto, conhecedor das regras.

Há, a este respeito, duas opiniões. É evidente que, se o jogo fosse uma situação da vida real, o jogador não poderia contar senão consigo próprio e teria de reagir da maneira mais adequada: neste caso a falta de vocabulário poderia muito bem ter influência no resultado do jogo. Mas a verdade é que muitos dos jogos são tão difíceis que não é necessário confundir o jogador desde o princípio, fazendo-o ter de descobrir o vocabulário antes de se poder debruçar minimamente sobre a resolução dos problemas.

Posto isto, se o autor for suficientemente amável para nos guiar no que diz respeito ao vocabulário, consideramos isso uma bonificação, e aceitamo-la com gratidão. Se não desvendar nada, encolhemos os ombros e continuamos o jogo. Contudo, pode ser particularmente irritante encontrar um pedaço de má programação (e, infelizmente, isto encontra-se em quase todos os programas — ressalvando um pequeno número de notáveis excepções). O jogador encontra um local que é descrito como segue: «*Você encontra-se numa clareira. À sua frente há uma cabana.*» Porém, o programa recusa-se a aceitar a palavra HUT (cabana), quando o jogador ordena «ENTER HUT» (entrar na cabana), respondendo: «*Não vejo nenhuma cabana aqui.*» É frequente que casos como este aconteçam: o problema reside no facto de o objecto ter sido incluído, de forma bastante traiçoeira, apenas como cor na descrição do local.

BREVE LÉXICO

Eis uma curta lista das palavras que o jogador encontra com mais frequência em jogos de aventuras. Esta lista não é exaustiva, de modo que, se as palavras aqui apresentadas não resultarem, procure outras num almanaque.

GO (ir): Esta palavra pode ser usada quando o jogador se encontra num local com uma única saída. No entanto, na maior parte dos jogos de aventuras é esperado um *input* de pelo menos duas palavras, e o computador pode perguntar: «*Go where?*» (ir para onde?) ou «*Try a direction*» (escolha uma direcção). Se na descrição inicial de um quarto está incluída uma janela, o jogador pode geralmente obter uma descrição da vista se introduzir «GO WINDOW» (ir à janela).

DIRECTION (direcção): É evidente que o jogador pode introduzir «GO NORTH», «GO WEST» «GO UP» (ir para norte, ir para oeste, ir para cima) e assim sucessivamente, mas a maior parte dos programas aceita simplesmente «SOUTH», «EAST» (Sul, Leste) ou até «S», «E». Isto poupa muito trabalho de dactilografia e torna o jogo muito mais rápido. Frequentemente, o pensamento paralelo pode fazer maravilhas quando o jogador está a decidir a direcção que vai tomar. Muitas aventuras, ao descrever um local, mencionam «Some of the exits are...» (algumas das saídas são...), o que permite ao jogador experimentar direcções que não são mencionadas na descrição. Além disso, é claro que os pontos cardeais principais não são os únicos que podem ser usados: por que não experimentar os poucos habituais «SW» ou «NE»? Se no local existir uma cornija, é frequentemente possível subir ou saltar para lá: geralmente isso compensa bastante, embora também não seja de estranhar que o jogador vá parar ao covil de um tigre devorador de homens.

Seja qual for o caminho que seguir, o jogador deve *fazer um mapa!* Se se perder, poderá retroceder ou recomeçar a aventura e chegar rapidamente ao local onde se encontrava antes de se ter perdido (ver SAVE — salvaguardar — mais abaixo).

QUIT (desistir): É esta a palavra que o jogador utiliza quando está pronto a desistir. Muito raramente, um autor poderá encarar esta

palavra como pertencendo ao pior dialecto californiano e exigir o uso de STOP (parar). Graças a Deus, este pedantismo é raro. Ao chegar a este ponto, na maior parte dos jogos de aventuras, o computador pergunta ao jogador se quer *de facto* desistir (QUIT), dando-lhe desta forma uma oportunidade de mudar de ideais.

Seguidamente, o computador apresenta a pontuação (*score*) do jogador, nos casos em que o programa possua este tipo de dispositivo. Nas aventuras de Scott Adams, o jogador é informado do número de tesouros que coligiu até esse ponto, sendo-lhe atribuída a posição baseada na comparação desse número com o número total dos tesouros disponíveis.

A editora Channel 8 Software, nas suas *Misterious Adventures,* contenta-se com o parar o jogo assim que o jogador tecla QUIT, não lhe sendo assim dada uma segunda oportunidade! Nas *Misterious Adventures* o jogador também não encontra uma rotina de *score* (pontuação). Se perguntar qual é a sua pontuação, é-lhe respondido «*This is not a football match!*» (isto não é nenhum jogo de futebol).

HELP (ajuda): Esta instrução dá origem a uma quantidade de respostas estranhas. O jogador tem de estar preparado para receber uns insultos valentes por admitir perante o computador que não faz a mais pequena ideia da maneira como há-de continuar. Muitos programas respondem simplesmente : «*Look around and try rephrasing your command*» (olhe à sua volta e tente modificar a sua instrução), o que significa que o jogador não vai receber ajuda absolutamente nenhuma durante todo o decurso do jogo. Outros dão, por vezes, uma pista cifrada, relativa a objectos ou seres que, nesse momento, se encontram no local. Seja como for, nenhum jogo de aventuras pode fornecer uma pista específica.

INVENTORY (inventário): Em qualquer altura que o jogador introduza esta instrução obterá uma lista das coisas que nesse momento transporta. A palavra pode ser abreviada para IN (o jogador não deve ficar surpreendido se o computador o interpretar mal e o levar para um recinto cheio de *orcs* sedentos de sangue!)[1]. Uma abreviatura provavelmente mais segura é, simplesmente, I.

[1] A partícula *in,* quando acompanha verbos de movimento tem o sentido de «entrar», o que explica a confusão feita pelo programa. *(N. da T.)*

LIST (lista): Esta instrução desempenha frequentemente a mesma função que INVENTORY.

LOOK (olhar): Esta instrução dá ao jogador outra oportunidade de ver a divisão ou o local em que se encontra.

REDESCRIBE (voltar a descrever): Tem o mesmo efeito que LOOK.

EXAMINE (examinar): Quando der com o corpo de um vagabundo morto ou com um monte de folhas — quer dizer, quando encontrar um objecto inanimado —, o jogador deve examiná-lo. A Regra de Ouro «*Tudo tem uma finalidade*» não deve ser esquecida. Os programas de aventuras não dispõem de espaço de memória suficiente para incluir elementos não essenciais, por isso é possível partir do princípio de que tudo o que vê tem uma finalidade no plano total. Assim, uma pilha de folhas mortas esconde muitas vezes uma espada refulgente, e num cadáver pode encontrar-se uma pulseira de diamantes.

SAVE/RESTORE (salvaguardar/reestabelecer): O uso judicioso desta instrução poupa muito tempo ao jogador. Antes de se lançar numa missão arriscada, como por exemplo entrar numa caverna escura ou abrir uma grande porta de madeira, o jogador deve salvaguardar (SAVE) a sua posição nesse momento. Assim, se lhe acontecer ser morto (o que é habitual), não terá de voltar a percorrer todo o caminho — caminho esse que foi, sem dúvida nenhuma, conquistado com dificuldade. RESTORE é a instrução que é normalmente requerida para que a posição anterior do jogador seja agora reintroduzida a partir de um disco ou fita.

PICK/TAKE/GET (apanhar): Uma destas instruções vai ser a que o programa aceita para apanhar um objecto. Para ver se de facto entrou na posse do objecto desejado, o jogador deve introduzir INVENTORY e verificar se ele foi acrescentado à lista de objectos que transportava.

DROP (largar): Inversamente, o jogador pode ter necessidade de deixar um objecto. A maioria dos programas tem um limite de peso que, realistamente, não pode ser excedido pelo jogador. Assim,

é requerido um cuidadoso compromisso entre a utilidade de um objecto e a capacidade que o jogador tem de o obter e transportar.

OPEN (abrir): Esta instrução é utilizada com muita frequência quando o jogador é confrontado com uma porta ou um receptáculo. A maioria dos programas é muito pretensiosa a este respeito. Habitualmente o jogador tem de voltar à chave da porta ou da arca antes de a poder abrir. Por vezes é necessário apanhar (PICK ou TAKE) a arca antes de ser possível abri-la.

KILL (matar): Esta é boa para os velhos entusiastas de *Dungeons & Dragons!* A maior parte das aventuras aceita esta palavra ou mesmo qualquer sinónimo como STAB (apunhalar), KICK (mandar desta para melhor), THROTLE (estrangular), BREAK (dar cabo de, partir) — esta última palavra também pode ser utilizada contra uma porta obstinadamente fechada — e por aí fora. Deixa-se à imaginação do jogador encontrar a palavra que vai originar a resposta desejada. Em todo o caso, é aconselhável que o jogador não se esqueça de, previamente, salvaguardar (SAVE) a sua posição.

D--- (insultos): Juntamente com todas as outras palavras em que o jogador pensa às três de manhã, quando está prestes a atirar pela janela fora o raio do computador, o programa compreende quase sempre alguns insultos. Se o não faz, isso não abona nada a favor do sentido de humor do programador; mas, por outro lado, também não se pode estar à espera que o programa reconheça todas as variações em que o jogador possa pensar.

Esta é, evidentemente, uma lista muito curta das palavras que o jogador vai encontrar. É possível que ele tenha de cavar (DIG), nadar (SWIM), voar (FLY), gritar (SHOUT), murmurar (WHISPER), rastejar (CRAWL), mexer-se (MOVE), cuspir (SPIT), desenhar (DRAW) e suicidar-se (COMMIT SUICIDE), entre muitas outras possibilidades. A única forma de descobrir quais são as instruções que um determinado programa aceita é experimentar todas as palavras que nos passam pela cabeça.

MONSTROS

Agora podemos divertir-nos um bocado e falar de alguns dos monstros que se podem encontrar num jogo de aventuras típico.

Para o nosso presente objectivo o termo «monstro» pode ser aplicado a qualquer ser interveniente no programa que possa fazer mal ao jogador, apesar de nem todos estes seres serem equacionáveis com a noção habitual de monstro.

Tal como acontece com os próprios programas, podemos considerar diferentes «categorias» de monstros. As aventuras clássicas, descendentes do jogo original *Adventures,* jogado em computadores principais, contêm monstros relativamente passivos, que têm tendência para ficar aonde estão, à espera que o jogador pense em algum estratagema brilhante que os assuste e os ponha em fuga, como aconteceu com a serpente da nossa cena no princípio deste livro. Outra possibilidade é o jogador ter de evitar o monstro e encontrar uma forma de passar por ele sem ser notado.

Os anões malignos das aventuras *Colossal Cavern* são uma notável excepção. De vez em quando aparecem e atiram uma arma ao jogador. O primeiro anão atira um machado, que o jogador deve apanhar e, subsequentemente, atirar aos anões seguintes. Todos eles lançam facas contra o machado inicial, mas se o jogador se lembrar de continuar a ir buscá-lo, não deve ter dificuldade em sobreviver aos ataques.

A inversão da Level 9 desta aventura clássica inclui uma parte final que acrescenta setenta locais ao jogo inicial. Aí o jogador pode realmente tirar a desforra desses anõezinhos! Deixando cair dinamite junto de uma multidão deles, é possível marcar muitos pontos. A propósito, no decurso desta sequência final, o jogador também pode marcar pontos se salvar da morte alguns elfos.

Originalmente, quando os jogos de aventuras eram praticados em computadores principais fora das horas de serviço, a impressora era frequentemente o único meio de saber o que estava a acontecer. Este facto impedia que o combate taco-a-taco, do tipo que se encontra em *D&D,* fosse exequível.

Isto leva-nos à próxima categoria de monstros: aqueles que encontramos em jogos de acção, ou jogos de *arcade*. Os monstros desta categoria são extremamente activos e andam decididamente a ver se apanham o jogador. O sistema de combate dos jogos de

representação de papel, como, *Dungeons & Dragons,* é frequentemente utilizado nestes jogos, sendo também muito popular num grande número de jogos de aventuras. Estes monstros são, de acordo com os antecedentes deste tipo de jogo, do tipo feroz, e sanguinário que se encontra frequentemente nos jogos de representação de papel. Segue-se uma lista destas adoráveis criaturas, juntamente com breves pormenores sobre os seus atributos e origens. Utilizando um critério muito subjectivo, os monstros foram colocados segundo uma ordem crescente da sua capacidade de aterrorizar. Assim, se o jogador encontrar um *Balrog,* deve normalmente tratá-lo com bastante mais respeito que, por exemplo, um *Orc.*

CÃES: Monstros de muito baixa extracção, mas muito malevolentes. Geralmente atacam aos pares. Para lutar com eles são suficientes as armas mais rudimentares.

LOBOS: Ainda mais malevolentes que os cães e, frequentemente, mais astutos.

ANÕES: Aparecem na versão original de *Colossal Cave,* bem como em todos os programas que se baseiam neste clássico. Geralmente, saltam das sombras, atiram um machado ou um punhal, e voltam a desaparecer. A maior parte das vezes não fazem mais que aborrecer o jogador.

ORCS: São o «homem dos sete ofícios» dos monstros, e aparecem, na maior parte das aventuras de tipo tolkienesco. Trata-se de criaturas extremamente desagradáveis, muito malevolentes e feias, que caçam em grupos, empunhando lanças ou cimitarras. Num dos jogos da Level 9 aparece um *Orc* archeiro.

WAUG: Esta criatura informe aparece no livro *The Hobbit* e também na aventura com o mesmo nome da Melbourne House.

SERPENTE: A maior parte dos programas que utilizam uma serpente fazem-no mais para criar um quebra-cabeças que para causar qualquer perigo físico (ver cap.1).

AVES: Encontram-se frequentemente no cimo das montanhas, sentadas sobre ovos de ouro. É melhor o jogador descobrir uma

forma de as assustar e as pôr em fuga antes de tentar apoderar-se do tesouro. As aves também são úteis contra as serpentes (ver cap. 1).

ELEMENTOS: Há-os de quatro tipos: Ar, Fogo, Terra e Água. É necessário que o jogador esteja na posse de magia de um tipo relacionado com o elemento antes de tentar lutar. A Electronic Arts utiliza elementos com grande sucesso no seu programa *Archon.*

VAMPIROS: Não é preciso dizer como é que se luta contra eles. Antes de os encontrar, o jogador deve ter apanhado, noutros locais, todos os objectos necessários. Se encontrar alho, não vire o nariz: pode ter a certeza de que há um vampiro por perto.

VAMPIROS DO ESPÍRITO: Estes pertencem a um tipo especial, que não quer o sangue do jogador...

FANTASMAS: Não é necessário uma espada ou uma lança para lutar contra eles!

LOBISOMENS: Contra eles só podem ser usadas armas mágicas, de preferência feitas de prata.

GOBLINS: Criaturas pequenas e muito feias que se deleitam em atormentar as suas vítimas, picando-lhes as pernas com paus aguçados.

HOBGOBLINS: Criaturas maiores, mais perigosas e mais astutas que os seus animalescos meio-irmãos.

HARPIA: Criatura alada, dotada de surpreendente força e agilidade.

SEREIA: Harpia marítima que se encontra geralmente a gozar o sol, cantando sobre rochedos perto do mar. A música *rock* nunca soou assim[1].

---

[1] Trocadilho entre os dois sentidos da palavra inglesa *rock,* que tanto pode designar um rochedo como um tipo de música moderna. *(N. da T.)*

TROLLS: Criaturas matreiras, ávidas e desajeitadas. Aguentam bem uma data de pancadaria, mas frequentemente não gostam muito da luz.

BARROW WIGHT: Aparições fantasmagóricas que povoam os desertos da Terra do Meio.

CENTAUROS: Criaturas com corpo de cavalo e tronco e cabeça de homem, armadas de arco e flechas que manejam com inexcedível maestria.

DIABRETES DE FOGO: Chamas pequenas e ágeis, que servem apenas para aborrecer o jogador.

GIGANTE DE FOGO: Labareda muito grande e perigosa.

LAGARTO-TROVÃO: É um dos mais potentes dos monstros. Não se meta com ele, a não ser que esteja bem armado com armas tradicionais e as maneje com bastante destreza.

VERME DA AREIA E VERME PURPÚREO: Extremamente perigosos! O verme da areia consiste basicamente numa boca que precede um estômago com 18 metros de comprimento; o verme purpúreo é semelhante, mas tem olhos e não está confinado a zonas arenosas.

MINOTAURO: É a bem conhecida criatura com corpo de homem e cabeça de touro. Perigosíssimo, como seria de esperar de um touro altamente inteligente movido por uma excitação furiosa.

WIVERN: Outra criatura alada, mas que vem equipada com terríveis colmilhos e garras afiadas como lâminas.

BALROG: É um dos monstros mais perigosos, muito querido dos escritores de jogos de aventuras.

DRAGÃO: É provavelmente o *mais perigoso* de todos os monstros. Só os mais fortes e mais valentes dos aventureiros devem aproximar-se dele, ou então os mais manhosos, pois os dragões nem sempre querem assar o jogador!

Claro que isto é apenas uma lista parcial dos monstros que o jogador pode vir a encontrar. A maior parte dos programas de aventuras inclui alguns destes e ainda outros que lhes são próprios. O escritor de programas de aventuras é limitado apenas pela sua própria imaginação. Qualquer coisa pode servir para um jogo: desde bonecos de neve até autocarros de dois andares, já vi ser utilizado tudo como inimigo.

A maior parte dos programas que se baseiam num sistema de combate no estilo de *D&D,* mantêm os jogadores informados a cada momento acerca do seu estado físico. Esta informação pode tomar a forma de pontos físicos, de combate ou de alimento. O jogador pode mesmo ser informado por meio de uma combinação destes três tipos de pontos, mas, apesar disso, ao jogar, sentirá claramente que a decisão de lutar ou não com um determinado monstro tem de ser tomada por si próprio, de acordo com a que possui, utilizando o sistema a que o programa está vinculado, juntamente com os conhecimentos que possui acerca da categoria a que o monstro pertence.

É evidente que o jogador pode não ser previamente avisado da presença de um monstro, e, assim, ter de lutar quer queira quer não. Este tipo de programa é muito pouco leal e não é comum: a maioria dos jogos dá ao jogador um certo conhecimento das capacidades do monstro e um método de fuga para o caso de ele desejar não aceitar o desafio. Ainda assim, a fuga pode não ser bem sucedida se o monstro for especialmente rápido!

FEITIÇOS

O tipo de combate de que temos vindo a falar neste capítulo, com o sistema de pontos de força ou de combate, está, evidentemente baseado no lado físico da luta, ou seja, o que interessa é que o jogador consiga dar alguns golpes bem calculados na cabeça do monstro com a sua grande espada ou outra poderosa arma. No entanto, muitos programas dão ao jogador a opção de utilizar poderes mágicos e colocar o monstro sob a acção de feitiços simples. Esta ideia está firmemente estabelecida em *D&D, T&T* e noutros jogos de representação de papel em que muitas páginas dos livros de

instruções são dedicadas a feitiços, já de si complicados, que se vão tornando cada vez mais complicados ainda à medida que a vida do jogador enquanto feiticeiro se vai alongando.

Porém, a ideia básica do feitiço não é diferente da anterior: dar cabo do monstro *(ZAP)* através de um feitiço bem calculado. Este pode tomar a forma de um simples feitiço de sono (SLEEP), que põe o monstro inconsciente durante um determinado período de tempo, ou de um dos feitiços mais complexos que se encontram em *Telengard,* da Avalon Hill — destinado a um grande número de aparelhos americanos entre os quais se contam o *Atari,* o *Commodore* e o *Apple* —, ou em *Velnor's Lair* (O Covil de Velnor), da Quicksilva.

Velnor's Lair trata do feiticeiro negro Velnor, que se escondeu no labirinto *goblin* de Mount Elk. A tarefa do jogador é evitar que Velnor se transforme num demónio na Terra, o que, a acontecer, não iria ser nada bom para a humanidade. O jogador deve explorar o labirinto, enfrentando os seus horrorosos guardas, e, finalmente, encontrar-se cara-a-cara com o próprio Velnor. O jogador começa a busca armado com uma tocha, um isqueiro de pederneira e um taco. No princípio do jogo é-lhe perguntado se ele quer ser:

UM GUERREIRO: Esta será, evidentemente, a escolha de quem goste de andar à pancadaria com monstros. O guerreiro não tem poderes para lançar feitiços, mas pode utilizar quaisquer elementos mágicos que encontre no labirinto.

UM FEITICEIRO: Esta escolha converte o jogador num alvo bastante fácil, pois dá-lhe capacidades físicas muito limitadas. A força do feiticeiro reside nos seus feitiços, que são três:

1) *Polymorph* (Polimorfismo): Este feitiço cifra-se na perda de um ponto de feitiçaria (o jogador inicia o jogo com um total de dez) e transforma num inofensivo sapo qualquer criatura viva desprovida de protecção mágica. A taxa de sucesso é, normalmente, de 50%.

2) *Teleport* (Teletransporte): Este feitiço cifra-se na perda de três pontos e coloca o feiticeiro na entrada do laririnto, juntamente com tudo aquilo que ele estiver a transportar.

3) *Fireball* (Bola de fogo): É o mais potente dos feitiços, mas também é aquele que tem um custo mais elevado, pois cifra-se em cinco pontos. Através dele podem ser mortas quaisquer criaturas que não gozem de protecção mágica no local em que ele for lançado.

UM SACERDOTE: Esta escolha coloca o jogador a meio caminho entre o feiticeiro e o guerreiro no que toca às suas capacidades. Tal como o feiticeiro, parte com um total de dez pontos e tem à sua disposição três feitiços.

1) *Shield* (escudo): Tem um custo de três pontos e torna o jogador menos vulnerável aos ataques; porém, só pode ser usado uma vez.

2) *Dispel undead* (afastar os que não estão mortos): Tem a função que o seu nome indica, e também custa três pontos.

3) *Heal* (curar). Custa quatro pontos e cura todos os ferimentos do sacerdote.

Após este longo preâmbulo, o jogo segue bastante à risca o modelo de *Wumpus.* O jogador explora um dédalo de túneis onde há muitos problemas que têm de ser resolvidos e, é claro, existe um grande número de monstros com os quais se pode lutar. Tirando as rotinas de feitiços, que são complexas e pelas quais vale a pena obter o programa (ainda que haja outros programas mais recentes com rotinas ainda mais complexas que estas), *Velnor's Lair* é uma aventura banal, mas que deve dar muito gosto aos entusiastas de *D&D.* Na altura em que este livro está a ser escrito, ainda se encontra nas colunas de ajuda a jogadores encalhados de todas as revistas, o que indica que, obviamente, passou o teste de resistência ao tempo.

CAPÍTULO 3

Outras aventuras

Neste capítulo vamos tratar de alguns outros grandes nomes dos jogos de aventuras nos Estados Unidos e no Reino Unido, e falaremos ainda do *software* destinado ao *QL* que se encontra actualmente no mercado.

Das produtoras de *software* americanas, há duas muito mais importantes que as outras na produção de aventuras em texto: a Adventure International, de Scott Adams, e a Infocom.

SCOTT ADAMS

O primeiro nome que a maioria das pessoas ouve quando começa a praticar a sério aventuras em computador é o de Scott Adams. Isto acontece particularmente se possuir um aparelho americano, tal como um dos *Commodore,* dos *Apple* ou dos *Atari.* Scott Adams e a sua companhia quase monopolizaram os jogos de aventuras americanos, entre os quais os jogos destinados a aparelhos britânicos, como o *Spectrum,* o *Dragon* e o *BBC,* se encontram muito nitidamente em minoria. Na altura da elaboração deste livro, estão a aparecer para todos estes aparelhos britânicos versões escritas por Brian Howarth, da Channel 8 (suspeitamos que a Adventure International teve de ser bastante espicaçada até se convencer de que havia realmente mercado para jogos adaptados a estes aparelhos!).

Foi apropriado ser Howarth a implementar estas aventuras para computadores britânicos, dado que os programas da Channel 8 têm muito em comum com as aventuras de Adams, chegando esta semelhança ao pormenor, já mencionado, da existência de dedicatória no início de cada jogo.

Scott Adams começou a sua carreira como programador de computadores na Florida e, tal como aconteceu a muitos programadores antes e depois dele, descobriu a versão original de *Colossal Cave,* escrita por Crowther e Woods, no computador principal da companhia. Fascinado pela aparição desse mundo fantástico, Scott Adams escreveu uma versão no seu *TRS-80* antes de lançar um programa comercial chamado *Adventureland* (Terra da Aventura), em 1978. O projecto quase morreu à nascença quando Alexis, a sua mulher, pôs o disco no forno. Felizmente para Scott e para o mundo agradecido, o forno não estava ligado!

Em seguida, Adams fundou a Adventure International, que conta hoje vários sucessos em jogos de *arcade,* incluindo o grande *Preppie,* bem como muitos programas de negócios e utilidades. A Adventure International está também a promover outro escritor de jogos de aventuras, Jyym Pearson, que já escreveu quatro aventuras excelentes.

Os doze programas escritos por Scott (e Alexis) Adams são reconhecidos como aventuras em texto clássicas.

Eis as doze aventuras:

Adventureland (Terra da Aventura): Esta aventura foi o começo de tudo. Há, ao todo, treze tesouros a serem encontrados e cerca de vinte e nove locais a serem reconhecidos. Todas as outras aventuras seguem o padrão desta. Adams escreveu mesmo um compilador de aventuras que funciona como uma estrutura onde qualquer argumento se pode apoiar.

Pirate's Cove (A Enseada do Pirata): Esta aventura mostra-nos a técnica posta em prática. De facto, foi escrita por Alexis Adams, utilizando o compilador. Esta é, provavelmente, a mais fácil das aventuras de Scott Adams, mas não pensem, nem por um momento, que qualquer delas deixa de ser frustrante e diabolicamente difícil. *Pirate's Cove,* que possui vinte e cinco locais ainda que só inclua dois tesouros, é uma aventura muito humorística.

Mission Impossible (Missão Impossível): Esta aventura é uma corrida contra o tempo, em que o jogador tenta impedir um sabotador de fazer explodir um reactor nuclear.

Voodoo Castle (O Castelo Vudu): Esta aventura, que se passa num castelo com vinte e quatro quartos, leva naturalmente à aventura seguinte.

The Count (O Conde): Neste caso, trata-se do conde Drácula. Ainda que esta aventura conte apenas dezanove locais, as coisas mudam, no decorrer do jogo, quando o dia dá lugar à noite.

Strange Odyssey (Estranha Odisseia): Esta aventura tem lugar no espaço, numa nave espacial avariada que se despenhou num planeta desconhecido. Há vinte e dois locais e apenas cinco tesouros, mas o último destes vai ser extremamente difícil de encontrar!

Mistery Fun House (A Misteriosa Casa dos Divertimentos), *Pyramid of Doom* (A Pirâmide da Morte) e *Ghost Town* (A Cidade-Fantasma): Estas aventuras constituem um trio cujos cenários, dados os títulos, são bastante óbvios. *Fun House* tem trinta e sete locais, e, para muita gente, é a preferida das doze aventuras. A segunda pode ser bastante complicada e imprevisível, e a terceira é extremamente difícil: tem perto de quarenta locais, e pode ser resolvida de mais de uma maneira. Nesta última aventura entra um par de objectos que são trazidos da pirâmide.

Savage Island (Ilha Selvagem), 1.ª e 2.ª partes: Estas duas aventuras formam um todo, pois a primeira tem de ser resolvida antes que a segunda possa ser empreendida. O objectivo da primeira é a obtenção de uma palavra que vai servir de senha na segunda. São as mais difíceis das aventuras de Scott Adams e não são para jogadores pouco corajosos.

Golden Voyage (Viagem Dourada): Esta aventura é menos difícil que as duas aventuras anteriores e trata da corrida que o jogador tem de empreender para trazer ao rei o elixir da juventude.

À primeira vista, estas aventuras podem parecer exactamente como quaisquer outras; contudo, elas foram as primeiras. De fac-

to, a maioria das outras aventuras em texto limita-se a imitar a série de Scott Adams. Ainda não há notícias sobre novas aventuras desta série, mas os programas existentes vão manter muitos jogadores felizes durante horas e horas, entretidos com as suas tentativas de resolver as charadas de Adams.

INFOCOM

Se Scott Adams é o Channel 8 americano, a Infocom é certamente a Level 9 americana. Tanto os programas da Infocom como a forma de os apresentar são muito espectaculares: basta ver o conjunto de objectos que é fornecido com *Deadline* (Data Limite). Esta aventura é uma espécie de *Cluedo* sofisticado, em que o jogador é o inspector, tentando resolver o mistério do assassínio. São-lhe concedidas doze horas para levar a cabo a tarefa, e é-lhe dada uma carga de papelada constituída por relatórios de laboratório, etc. Também lhe é fornecida uma página de jornal, um bilhete rabiscado e uma carteira de fósforos com um número de telefone escrito atrás. Tudo isto intensifica a atmosfera que rodeia a aventura e é típico da espectacularidade com que a Infocom dota os seus programas.

Mas vamos começar pelo princípio. A Infocom é a primeira de uma nova linhagem de companhias que revolucionaram o mercado dos jogos de aventuras. É uma companhia jovem, formada por jovens profissionais de computadores que trabalham desde muito novos entre as paredes sacralizadas do MIT[1].

A Infocom desempenhou por si só um grande papel no desenvolvimento dos jogos de aventuras. O primeiro programa desta editora, *Zork,* tornou-se um clássico apenas igualado pelo programa original *Adventures* e pela série de Scott Adams. Ainda que tivessem escrito apenas este programa, os seus autores teriam conquistado um lugar seguro no Salão das Celebridades das Aventuras. Mas eles lançaram, desde então, *Zork II* e *III,* estando qualquer destes programas mais que apto a competir com o primeiro.

[1] MIT, Massachussets Institute of Technology, prestigiadíssima escola superior norte-americana. *(N. da T.)*

Zork I foi originalmente escrito em MDL, uma linguagem inspirada na LISP, em computadores principais. Nos princípios da década de 80, a Infocom usou a sua competência para implementar o programa em microcomputadores. Esta aventura baseia-se muito no jogo original de Crowther Woods, *Colossal Cave.*

Estes jogos encontram-se agora disponíveis para um largo leque de computadores, incluindo os *Apple,* os *Commodore,* os sistemas *CP/M,* os *Tandy,* os *Osborne* e os *DEC.*

Uma das razões que justificam o sucesso da Infocom é a linguagem utilizada nos seus programas. As descrições não são sumárias, como em outras aventuras: são, pelo contrário, extremamente espirituosas e ricas em pormenores. Enquanto as aventuras de Scott Adams são bastante avaras nos cenários e apenas aceitam instruções curtas da parte do jogador, contentando-se com fornecer-lhe quebra-cabeças cada vez mais difíceis, a Infocom desenvolveu um «analisador sintáctico» extremamente sofisticado, uma rotina que permite ao jogador comunicar com o computador numa linguagem natural.

Assim, uma aventura de Scott Adams diria: «*Estou num andar em Londres. Os elementos visíveis são...*» O jogador deve replicar com uma ou duas palavras, por exemplo «*Apanhar ténis*». Por seu lado, uma aventura da Infocom diria (note-se a mudança de ênfase, pois este é um jogo na primeira pessoa): «*Você está de pé ante um precipício...*» (a descrição pode perfeitamente continuar até encher um visor inteiro!). A isto o jogador pode replicar, por exemplo: «*Atirar o jornal, o livro vermelho e a revista para o precipício*» ou, correspondentemente, «*Apanhar livro. Ir para Norte. Deixar cair campainha, livro e vela*». É possível pedir a outras personagens que executem diversas acções ou divulguem certas informações como, por exemplo, a localização do ouro. O programa tenta mesmo decifrar o que o jogador quer dizer quando não dá informação suficiente.

LOOK (olhar) é uma instrução admitida em todos os jogos de aventuras, embora possa tomar significados diferentes. Mas pensem só nas possibilidades que se nos abrem em *Zork,* em que LOOK INSIDE (olhar para dentro de) e LOOK UNDER (olhar para debaixo de) também são instruções válidas!

Há várias instruções que podem ser usadas quando se viaja pelo mundo de *Zork.* Por exemplo, VERBOSE (verboso) assegura que seja dada ao jogador uma descrição muito pormenorizada de

A.V. 107 — 5

cada divisão de cada vez que o jogador aí entra. BRIEF (breve) dá-lhe apenas a descrição das divisões em que se encontra pela primeira vez, e SUPER BRIEF (superbreve) proporciona apenas o nome de cada uma delas.

Ainda que cada um dos três *Zork* possa ser jogado e apreciado independentemente dos outros, há em cada um deles vislumbres do resto da série, e, de facto, os argumentos estão encadeados. Em *Zork I,* o Grande Império Subterrâneo confronta o jogador com situações delicadas que podem ir do tipo místico ao tipo macabro, havendo vinte tesouros a ser encontrados antes que o jogador tenha de fugir. *Zork II* leva o jogador a novas profundezas do reino subterrâneo, onde ele vai encontrar o Feiticeiro de Frobozz. No jogo final da trilogia, o jogador encontra o próprio Mestre Construtor das Masmorras, que tem nas mãos o seu destino.

Apesar de não terem rival, as aventuras *Zork* são apenas a ponta do icebergue da Infocom.

Já falámos de *Deadline.* Esta aventura é completada por *The Witness* (A Testemunha), que é um policial com raízes nas novelas de crime populares dos anos 30 e 40. Este programa também arvora um processo recheado de pistas sob a forma de carteiras de fósforos, jornais, e por aí fora.

A aventura de ficção científica *Starcross* (Através das Estrelas) passa-se no ano 2186. O jogador tem de se encontrar com uma nave estelar dos limites exteriores da galáxia, e de entrar no seu misterioso interior. Uma vez aí, encontra seres de outros mundos, uns benignos, outros malignos. Outra aventura de ficção científica é *Planetfall* (A Queda no Planeta), que está escrita com um espírito mais ligeiro que as outras. O jogador é um náufrago espacial num planeta distante, que está armado apenas com uma escova com múltiplas aplicações, proveniente da patrulha, e tem a ajuda de um companheiro, um endiabrado robot. O jogador é colocado perante o desafio de salvar o mundo condenado e devastado pela peste enquanto tenta impedir-se de desatar a rir.

Entre as aventuras da Infocom, a mais original é *Suspended* (Suspenso), que tem como subtítulo «*Um pesadelo criogénico*». Da autoria do escritor contratado Michael Burlyn, um conhecido criador de romances de ficção científica, a aventura trata de suspensão criogénica: a suspensão da vida pelo frio. Ao acordar de um longo sono gelado, o jogador encontra o planeta em crise e lembra-se de que lhe tinha calhado na rifa, quinhentos anos antes, servir como


```
>ALL ROBOTS,REPORT LOCATION
FC: Cryolink established to Whiz.
WHIZ: In the Weather Monitors.

FC: Cryolink established to Poet.
POET: In the Weather Monitors.

FC: Cryolink established to Auda.
AUDA: In the Sloping Corridor.

IRIS INTERRUPT: You may not be aware
of this, but I'm not working. I need
to get repaired.

FC: Cryolink established to Sensa.
SENSA: In the Weather Monitors.

FC: Cryolink established to Waldo.
WALDO: In the Weather Monitors.

FC: Cryolink established to Iris.
IRIS: In the Weather Monitors.
```

Fig. 3.1(*a*) — *«Suspended» da Infocom. As seis funções robóticas do jogador informam sobre as suas localizações iniciais.* Iris *interrompe com um lamentoso pedido de ajuda*[1].

---

[1] TODOS OS ROBOTS, INFORMAR LOCAL
FC: Crioligação estabelecida com *Whiz.*
WHIZ: Nos indicadores meteorológicos.
FC: Crioligação estabelecida com *Poet.*
POET: Nos indicadores meteorológicos.

FC: Crioligação estabelecida com *Auda.*
AUDA: No corredor escorregadio.

IRIS INTERROMPE: Talvez não tenha dado por isso, mas não estou a funcionar. Preciso de ser arranjado.

FC: Crioligação estabelecida com *Sensa.*
SENSA: Nos indicadores meteorológicos.

FC: Crioligação estabelecida com *Waldo.*
WALDO: Nos indicadores meteorológicos.

FC: Crioligação estabelecida com *Iris.*
IRIS: Nos indicadores meteorológicos.

```
>SENSA,REPORT
FC: Cryolink established to Sensa.
FC: Full report from SENSA
SENSA: Internal map reference --
Central Chamber
All around me charges flow, shaped by
the very nature of this room. The
electrons are being channeled into an
electrical column, central to this
environment.
In the room with me is Poet.
SENSA: I cannot sense a thing in my
grasping extensions.

>POET,REPORT
FC: Cryolink established to Poet.
FC: Full report from POET
POET: Internal map reference --
Central Chamber
It hops and skips and leaves a bit,
and can't decide if it should quit. It
tells the world what it should know,
but doesn't know when it's been shown.
In the room with me is Sensa.
POET: As far as I know, I'm Zen on
inventory.
POET: Sensory pads detect no abnormal
flow.
```

Fig. 3.1 (*b*) — «*Suspended*». Sensa *e* Poet *vêm a câmara central de formas muito diferentes*[1].

---

[1] SENSA, informar
FC: Crioligação estabelecida com *Sensa*.
FC: Relatório completo de SENSA.
SENSA: Mapa de referência interno.
Câmara central.
À minha volta flutuam cargas a que a própria natureza desta sala dá forma.
Os electrões estão a ser canalizados para uma coluna eléctrica que é o centro deste ambiente.
Poet está comigo nesta sala.
SENSA: Não consigo sentir nada nas minhas extensões tácteis.

POET, INFORMAR
FC: Crioligação estabelecida com *Poet*.
FC: Relatório completo de *POET*.
POET: Mapa de referência interno.
Câmara central.
Isto salta e saltita e pára um bocado, e não é capaz de decidir se há-de parar

Mentalidade Central do Planeta Contra. Utilizando um tabuleiro e peças de jogo, o jogador deve manobrar seis robots para tentar pôr termo à crise.

Uma vez que cada um dos robots tem acesso a um tipo específico de informação, a imagem geral tem de ser construída a partir do *input* que é recebido separadamente de cada robot. O robot *Poet* (Poeta) é, como o seu nome deixa adivinhar, dado a tiradas palavrosas e obscuras, enquanto os outros robots comunicam as suas informações de forma mais directa. O jogador também tem o controlo de *Waldo,* que é capaz de manipular objectos, e de *Wiz,* que pode interactuar com o computador central. Assim, no jogo *Suspended* não basta dizer «*Apanhar objecto*». O jogador tem de se assegurar primeiro da presença de *Waldo* no local depois de ter a certeza de que *Iris,* o robot que vê, já viu o objecto. Para além de tudo isto, é evidente que também pode acontecer que os robots se avariem.

Não há necessidade de fazer um mapa dos cinquenta e oito locais, como acontece na maioria das outras aventuras, pois o tabuleiro de jogo e as peças de borracha de que já falámos suprem essas funções. Há quatro níveis de jogo, um dos quais vai dar ao jogador a possibilidade de adaptar a aventura aos seus gostos pessoais. Os robots podem ser colocados em qualquer posição inicial e os vários sistemas ajustados.

Estas são as aventuras mais originais e estimulantes que conhecemos!

---

de vez. Diz ao mundo o que devia saber, mas não sabe quando é que lhe foi mostrado.
Sensa está comigo nesta sala.
POET: Tanto quanto sei, os meus inventários são Zen.
POET: As minhas extremidades sensoriais não detectam nenhum fluxo anormal.

O HOBBIT[1]

É provável que *The Hobbit* (O Hobbit) seja entre as aventuras para computador a mais bem conhecida, se não por experiência pessoal, pelo menos de reputação. É um clássico, e é uma das aventuras sobre as quais mais se tem falado e matutado! Segundo a verdadeira tradição de progresso do *software,* vão aparecer outros programas que, indubitavelmente, hão-de trazer melhoramentos relativamente a *The Hobbit,* mas este vai manter a sua posição na história das aventuras para computador.

O livro chamado *The Hobbit,* de J. R. R. Tolkien, talvez possa ser considerado como o prefácio da monumental obra *The Lord of the Rings* (O Senhor dos Anéis)[2], que é, provavelmente, a mais rica fonte de material para o escritor de jogos de aventuras. Este vasto trabalho, com os seus três volumes de narrativa absolutamente fascinante e historicamente apoiada em *The Silmarillion (O Silmarillion)*[2], juntamente com a informação enciclopédica que nos é dada por autores como Tony Tyler no seu *Tolkien Companion* (guia da obra de Tolkien), constitui uma verdadeira mina de ideias. Foi um golpe de génio da Melbourne House o facto de ter adquirido, do legado de Tolkien, os direitos de utilização de *The Hobbit* e ter produzido um programa que não fica atrás do livro. *The Lord of the Rings,* que é a continuação lógica de *The Hobbit,* é o programa de que toda a gente está à espera.

A concepção básica desta aventura já tinha sido usada várias vezes antes de ser utilizada em *The Hobbit.* É uma aventura em texto, escrita segundo as boas tradições, à qual é dado apoio visual. A Melbourne House escolheu muitas cenas do livro, que ilustrou com lindas gravuras de alta resolução que aparentam ser obra da impressão de um artista.

Isto representa um progresso lógico relativamente aos programas anteriores de outros autores, que incluíam apoio visual menos sofisticado.

Os outros pormenores importantes que separam *The Hobbit* das anteriores aventuras com texto e imagens são aquilo a que a Melbourne House chama «animação» e «*Inglish*»[1]. Neste contexto, animação significa que as personagens do programa, tais como Thorin e Gandalf, os vários monstros, como Gollum e o Mordomo (a propósito: não foi ele o culpado), e os elfos da floresta, vão vivendo a sua vida enquanto o jogador, no papel de Bilbo Baggins, continua com a aventura. Na prática, isto significa que as personagens vão aparecendo em cena e desaparecendo de novo, entoando cantigas acerca de ouro e assim por diante. Até já aconteceu Gandalf levar a porta de entrada de *Bag End*[2]! Estas coisas não são necessariamente úteis, mas, apesar disso, são divertidas. *Inglish* (não é gralha) é o nome que a Melbourne House dá às frases compostas.

Imaginemos que o jogador alcançou o covil do dragão. Este encontra-se, como é evidente, de guarda ao ouro, esse ouro pelo qual o jogador arrostou com tantos perigos, a fim de o resgatar das garras do monstro. Quando o jogador chega, quer erguer a espada, lutar com o dragão e, depois, apanhar o ouro. A maioria dos programas exigiria que o jogador introduzisse várias instruções antes

```
You are in a comfortable tunnel like hall
To the east there is the round green door
You see:
     the wooden chest.
     Gandalf.
     Thorin.
Gandalf opens the round green door.
Thorin waits.
```

Fig. 3.2(*a*) — *«The Hobbit», Melbourne House. Eis como tudo começa*[3].

[3] Você está numa confortável sala em forma de túnel. A Leste encontra-se a porta redonda verde.
Você vê:
a arca de madeira
Gandalf
Thorin
Gandal abre a porta redonda verde.
Thorin espera.

[1] Para conhecer a história, consultar a obra que se encontra na nossa col. «Século XX». *(N. do E.)*

[2] Qualquer destas obras está incluída na nossa col. «Século XX». *(N. do E.)*

[1] Grafia deturpada de *english*, inglês. *(N. da T.)*

[2] A residência de Bilbo Baggins. *(N. da T.)*

```
You go east.
You are in a gloomy empty land with dreary
hills ahead
To the west there is the round green door
Visible exits are: east north northeast
You see:
     Nothing
Gandalf enters.
Thorin enters.

You go east.

You are in the trolls clearing
Visible exits are: southwest southeast
north
You see:
     the hideous troll. The troll is
     carrying
       the large key.
     the vicious troll.
Gandalf enters.
Thorin enters.
The hideous troll says " Blimey, looks at
this!! Can yer cook'em? ".
The vicious troll says " Yer can try, but
he wouldn't make above a mouthful ".
```

Fig. 3.2(*b*) — *«The Hobbit». E eis como tudo poderia acabar*[1].

---

[1] Você vai para leste.

Está numa terra vazia e desolada, e à sua frente encontram-se montes tenebrosos.

A Oeste está a porta redonda verde.

As saídas visíveis são: Leste, Norte, Nordeste.

Você vê:

Nada.

Gandalf entra

Thorin entra

Você vai para Este

Você está na clareira dos *trolls*

As saídas visíveis são: sudoeste, sueste e norte

Você vê:

o *troll* medonho. O troll transporta:

a chave grande.


```
Day dawns.

You go west.
You are in in a clearing with two stone
trolls
Visible exits are: southwest southeast
north
You see:
     the large key.
Thorin enters.

You take the large key.

You are carrying
     a curious map.
     the large key.
     some lunch.
Thorin sits down and starts singing about
gold.
```

Fig. 3.2(*c*) — *«The Hobbit». Leia o livro, e deve conseguir desembaraçar-se sem problemas. Thorin ainda vai voltar a cantar cantigas sobre ouro*[1].

---

o *troll* malevolente.

Gandalf entra.

Thorin entra.

O *troll* medonho diz: Ena pá, já vistes isto? Achas qu'a gente o p'dia c'zinhar?

O *troll* malevolente diz: Lá p'der pode-se. Mas nem dava p'á cova dum dente.

[1] O dia nasce.

Você vai para Oeste.

Está numa clareira com dois *trolls* de pedra.

As saídas visíveis são: Sudoeste, Sueste e Norte.

Você vê:

a chave grande

Thorin entra.

Você agarra a chave grande.

Você está a transportar:

um mapa curioso

a chave grande

alguma comida

Thorin senta-se e começa a entoar uma canção sobre ouro.

de o objectivo final ser atingido. Assim, poderia muito bem ser necessário escrever: DRAW SWORD (desembainhar a espada), KILL DRAGON (matar o dragão) e, finalmente, GET GOLD (apanhar o ouro).

Contudo, o programa *The Hobbit* aceita instruções do tipo «*Draw sword and kill dragon, then get gold and leave*» (Desembainhar a espada e matar o dragão, e em seguida apanhar o ouro e ir embora), o que poupa cansaço dos dedos e dá, além disso, uma sensação de maior naturalidade.

Esta aventura pode ser resolvida de muitas maneiras. Ao contrário do que acontece nos programas tradicionais, que só admitem uma solução para o quebra-cabeças completo, há vários métodos possíveis de completar *The Hobbit*. Assim, qualquer que seja a forma escolhida pelo jogador para resolver o quebra-cabeças final, espera-o um jogo diferente se ele quiser jogar de outra maneira. Contudo, o objectivo do jogador mantém-se: resgatar o ouro que está na posse de *Smaug*, o dragão, e voltar para o conforto da sua casa, sob a colina, em *Bag End*. Ao longo do jogo vai sendo atribuída ao jogador uma pontuação percentual. Há pessoas que terminaram a aventura com pontuações muito baixas, como 42%, e outras com pontuações muito altas, como 210%!

Como um aparte, gostaria de dizer que me parece que *The Hobbit* é um clássico, mas, apesar de tudo, um clássico com imperfeições. Por exemplo, muitas pessoas referiram que, premindo simultaneamente os botões CAPS SHIFT e 1 (o que pode perfeitamente acontecer por acaso) se bloqueia o programa. O mesmo destino espera os jogadores que tentarem entrar em certos locais sem estarem a transportar um determinado elemento. A animação (é assim que a Melbourne House chama à automotivação das personagens) manifesta-se de vez em quando de forma bizarra. Se matarem Gandalf (com toda a certeza não seriam tentados a fazê-lo!), ele pode muito bem aparecer, vivo e de boa saúde, numa data posterior (e pode, além disso, vir a transportar a sua porta de entrada...). Estas são pequenas peculiaridades a que os hobbitistas já se habituaram de tal forma que há quase um clube de caçadores de precalços no programa *The Hobbit*.

Apesar de ser superficialmente, uma aventura tradicional, *The Hobbit* proporciona um jogo com características únicas. Os quebra-cabeças apresentados por programas do estilo clássico (programas como os da Level 9) têm, geralmente, uma solução que per-

manece sempre na mesma. Uma vez que a aventura tenha sido completada, acabou-se. Claro que os quebra-cabeças podem levar meses a resolver, e vários programas incluem vastas áreas, que agem como uma espécie de diversão, constituídas por todos os locais que, ficando fora da linha condutora do jogo, não tomam parte na solução do problema mas servem para entreter o jogador. No próprio programa *The Hobbit* existe um intrigante local que está, como o computador informa o jogador, «demasiado cheio para que se consiga entrar.» Será que isto quer dizer que a memória do *Spectrum* está demasiado cheia para que os programadores possam incluir mais locais? É quase certo que não, pois eles são de certeza mais inventivos que isso! Haverá, o que é mais provável, um plano em preparação na Melbourne House para lançar uma continuação que, partindo desse local, entre na área não referenciada, à moda da Level 9?

A maior parte dos entusiastas dos jogos de aventuras de computador vai ficar satisfeita ao saber que a Melbourne House obteve os direitos de *The Lord of the Rings*. É quase certo que o programa vai aparecer em três partes, como o livro, se bem que seja tentador pensar que o *QL*, com os seus *microdrives*, podia ser um bom aparelho para ser abençoado com o novo programa.

O programa *The Hobbit* tem sido longamente discutido, e espero que, no futuro, continue a sê-lo. Se o leitor tem andado a pensar se há-de ou não comprar este programa encantador, frustrante, irritante e lindo, pode ter a certeza de que ele lhe vai proporcionar muitas horas de boa prática de aventuras.

«VALHALLA»

Se bem que *The Hobbit* tenha sido o primeiro dos jogos de aventuras «clássicos» para o *Sectrum* e tenha posteriormente sido traduzido para vários outros microcomputadores, incluindo os aparelhos *BBC* e *Oric*, há um par de programas mais recentes que merecem ser mencionados.

O programa *Valhalla*, da Legend, é uma aventura animada. O visor está dividido em duas áreas: um grande, para imagens, que ocupa dois terços da área total, e outra mais pequena, para texto, situada por baixo da primeira. Está associada a cada local uma

Fig. 3.3 — «*Valhala*», *da Legend. Vistas representativas do cenário, incluindo um ou dois habitantes, extraídas da versão para o* «*Commodore 64*»[1].

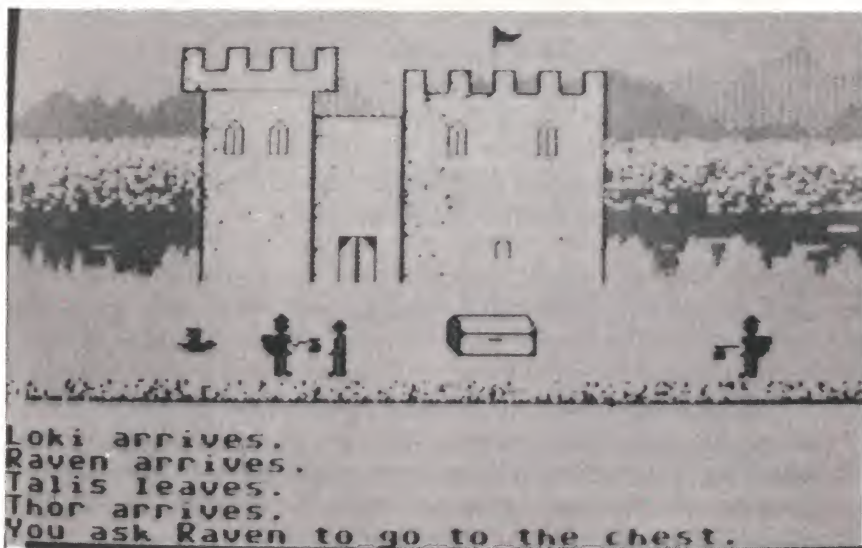
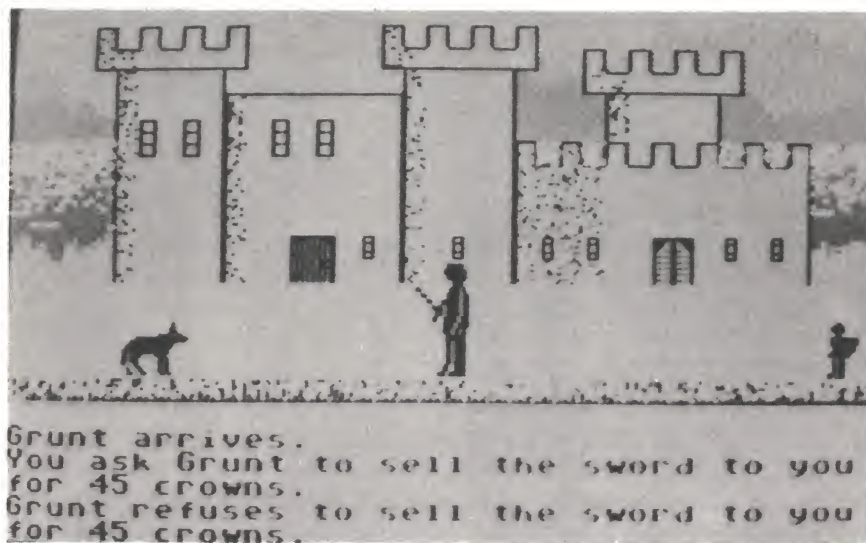



imagem construída a partir de um determinado número de elementos estandardizados. Há muitos castelos, com bandeiras esvoaçantes, e também desertos gelados. Quando a imagem acaba de ser constituída a partir de vários «blocos», são colocadas as personagens, entre as quais se encontra sempre o jogador, como é evidente.

Valhalla trata das seis demandas do jogador em busca de seis objectos (a chave *Ofnir,* o anel *Drapnir,* etc.). Uma vez que tenha tido êxito nas suas demandas, o jogador atinge Valhalla, a morada dos deuses. Enquanto se desloca pela paisagem, o jogador pode equipar-se com elementos básicos como vinho, comida, armas e armaduras. Como já deixámos entrever, o jogador não se encontra sozinho: há deuses maus (Loki e Hel, por exemplo) e deuses bons (Thor e Saga, por exemplo), bem como vários anões, gigantes, lobos e dragões, cada um dotado de características próprias.

São admitidas muitas das instruções habituais nos jogos de aventuras como LIST, GET, TAKE e HELP, e ainda algumas menos vulgares, como WHO (quem) e WHERE (onde); estas instruções revelam ao jogador quem é que, além dele, se encontra no local, e onde é que ele se encontra exactamente.

Outras características pouco habituais desta aventesma são o método de viajar (geralmente seguindo uma das direcções dos pontos cardeais, mas por vezes através de *ringways,* caminhos dos anéis, que só podem ser utilizados se se possuir um anel e, além disso, não se sabe sempre aonde é que se vai parar quando se utilizam...), e as formas de influenciar as personagens para que elas façam o que o jogador quer (por exemplo, lutarem contra outra personagem ou cederem uma arma ou comida). Melhor ainda, cada personagem age de acordo com os seus próprios traços de carácter, mas o jogador pode tornar uma personagem mais cooperante se atacar outra personagem com traços de carácter opostos. As-

---

[1] Primeira imagem:
Grunt chega.
Você pede a Grunt que lhe venda a espada por quarenta e cinco coroas.
Grunt recusa-se a vender-lhe a espada por quarenta e cinco coroas.
Loki chega.
Corvo chega.
Talis parte.
Thor chega.
Você pede ao Corvo que se dirija à arca.

sim, o bonzinho Odin estará mais inclinado a ajudar o jogador se este atacar Hel e Gripe, que são um par de malandros.

O programa está escrito essencialmente em BASIC, tendo algum código-máquina para ajudar a desenhar as imagens, e a acção é, por vezes, bastante lenta (ainda que a versão para o *Commodore 64* tenha sido um pouco acelerada). Apesar disso, o jogo é fascinante. De facto, a Legend prefere chamar-lhe um «filme para computador», dado que é perfeitamente possível o jogador sentar-se frente ao visor e ficar quieto a ver o que acontece.

«LORDS OF MIDNIGHT»

A editora Beyond Software lançou um par de jogos *arcade* antes de *Psytron*. Este jogo é um trabalho complexo — de facto, provavelmente complexo de mais para o seu próprio bem; no entanto, vendeu-se muito bem, a julgar pelos índices de venda por ele atingidos. Uma ou duas das técnicas introduzidas no jogo são inovadoras (por exemplo, o método de avançar de um nível para o seguinte num ordenamento percentual baseado numa média de melhores pontuações). O programa seguinte desta editora é bastante duro de roer.

Lords of Midnight (Os Senhores de Meia-Noite) é diferente de qualquer outro programa de aventuras lançado até agora para qualquer dos aparelhos existentes. A brochura muito bem apresentada que acompanha o programa contém cinco capítulos do *Book of Midnight* (o Livro de Meia-Noite). Este extracto conta os inícios da Guerra do Solstício, fala de Morkin e de Luxor e da sua luta contra o maléfico Doomdark.

Em *Lords of Midnight* não há imagens móveis. As cenas em cada local são, no entanto, espectaculares. Tal como em *Valhalla,* são constituídas a partir de vários blocos estandardizados, mas cada uma parece nova. A contracapa da brochura contém um mapa da Terra da Meia-Noite, e o jogador pode movimentar-se livremente através da paisagem. Quando o jogador chega a um novo ponto, pode olhar à sua volta, na direcção dos pontos cardeais, e aquilo que vê é apresentado no visor; se avançar os elementos em direcção aos quais se movimenta aumentam de tamanho.

A missão do jogador é, simplesmente, destruir Doomdark, que

é apoiado pelo fiel Spectrum. A vitória pode ser assegurada de duas maneiras: pode ser uma vitória estratégica, em que a fortaleza de Ushgarak é tomada e invadida pelas forças comandadas pelo jogador; ou então Morkin pode encontrar a Coroa de Gelo e destruí-la.

Para realizar tudo isto, o jogador pode chamar quatro personagens. A primeira, e a mais importante delas, é Morkin, o filho de Luxor. A sua tarefa é capturar a Coroa de Gelo, mas para que ele possa fazê-lo é necessário que lhe seja permitido alcançar o seu objectivo. O jogador pode ajudá-lo criando manobras de diversão com o seu exército. Luxor é quase tão importante quanto o seu filho, pois usa o Anel da Lua, o único escudo eficaz contra o Medo Gelado que desmoraliza e depaupera as forças de qualquer dos Livres que não esteja sob a sua protecção. Luxor é o chefe de todos os exércitos, mas primeiro tem de influenciar os comandantes desses exércitos para os levar a abraçarem a causa. Corleth, *o Clarividente* e Rorthron, *o Sábio* completam o quarteto que o jogador pode dirigir a partir do teclado.

Quando se escolhe uma dos quatro personagens, o visor mostra-nos o que se apresenta a seus olhos, assim como o seu nome e brasão, cuidadosamente reproduzidos com toda a sua pompa heráldica, e diz para onde é que está a olhar. Por exemplo: «*Rorthron* o Sábio. *Está nas montanhas de Ahimar, olhando para Leste.*» Também pode estar presente um representante dos vários seres desagradáveis incluídos nesta aventura, tais como lobos, dragões e os empecilhos locais, os *Skulkrin.* O jogador pode ir fazendo a personagem girar de acordo com os oito pontos cardeais e colaterais, sendo a vista correspondentemente diferente. Seguidamente, o jogador pode movimentar a personagem na direcção para a qual ela está voltada, ou decidir pensar (THINK). Se ele der esta instrução, a imagem desaparece do visor e é-lhe fornecida a informação acerca dos exércitos que a personagem pode controlar, do seu estado físico, etc. O jogador também pode escolher (CHOOSE). Dada esta instrução, é-lhe apresentada uma lista de opções que são afectadas pela personalidade básica da personagem.

Cada personagem pode ser movimentada até um número máximo de oito vezes antes que a noite caia. Sob a capa da escuridão movimentam-se as forças de Doomdark, e a batalha dá-se do outro lado do mapa, no local onde as forças opostas se encontram.

Lords of Midnight é um jogo extremamente complexo, que leva muitas horas a ser jogado. Um dos atractivos que apresenta é

Fig. 3.4 — «*Lords of Midnight*», *da Beyond. Belos instantâneos de alguns dos Senhores*[1].

o facto de ser diferente de cada vez que se joga. Parece-me ser a combinação ideal do jogo de estratégia e de aventuras, sendo o elemento de estratégia talvez preponderante, embora seja mais difícil chegar ao fim escolhendo a estratégia que a aventura. Estão planeados dois jogos que vêm na sequência deste: *Doomdark's Revenge* (A Vingança de Doomdark), que decorre nas terras que ficam a Nordeste de Meia-Noite, e *The Eye of the Moon* (O Olho da Lua), que se passa nas terras a Sul de Meia-Noite. Assim, os entusiastas de *Midnight* têm asseguradas, no futuro, muitas horas felizes.

AVENTURAS PARA O «QL»

Neste momento, como é evidente, não há de forma alguma muito *software* para o *QL,* ainda que eu tenha a certeza de que na altura em que este livro for lido já se vai poder escolher entre uma bela selecção de programas. Agora, como *software* disponível, existem apenas umas poucas de utilidades sobrevalorizadas (a instrução LOAD numa só tecla, uma lista tabelada em *microcartridge,* etc.), um pacote comercial muito dispendioso, e dentro do que nos interessa, duas aventuras.

West (Oeste) e *Zkul* são ambas da mesma editora de *software,* a Talent Computer Systems, de Glásgua, e, em conjunto, ilustram duas facetas do futuro panorama do *software* para o *QL.*

Quando os anteriores aparelhos *Sinclair* apareceram, os programadores começaram as suas experiências utilizando apenas uma folha de papel em branco. Para o *ZX81* não havia nenhuma fonte de inspiração. Por isso, os utilizadores deste aparelho pioneiro puderam pôr em prática ideias e técnicas novas. O *ZX81,* provido de uma linguagem BASIC mais poderosa, oferecia mais facilidades, mas o *software* disponível comercialmente levou alguns meses até atingir um volume considerável.

---

[1] Primeira imagem:
CORLETH O CLARIVIDENTE.
Encontra-se na Torre da Lua, a olhar para Norte, para as Colinas das Sombras.
MORKIN.
Encontra-se nas Colinas das Sombras, a olhar para Sul, para a Torre da Lua.

A.V. 107 — 6

Foi o *Spectrum* que modificou a etiqueta de passatempo que era atribuída, nesta altura, à computação. Com os conhecimentos que tinham sido adquiridos através do conjunto de instruções *Z80,* os programadores foram capazes de dominar muito rapidamente o novo aparelho, no qual a cor e a (mais) alta resolução das imagens conduziam a programas cada vez mais excitantes. É bem verdade que os primeiros meses foram dedicados a versões de *Othelo, Startrek, Hammurabi* (a maior parte deles convertidos do *ZX81*) e a numerosos derivados de *Space Invaders, Scramble* e *Moonlander,* que mantiveram o público satisfeito durante este período. No entanto, 1984 vai ser recordado como o ano em que os jogos de *arcade* para computadores pessoais se desenvolveram realmente, tornando-se um grande negócio; e os autores dos novos jogos responderam levando o *Spectrum* ao limite máximo das suas capacidades.

Terão esses limites sido realmente atingidos? Bom, é perigoso ser peremptório acerca de seja o que for quando se está a falar de computadores pessoais, mas estamos convencidos de que é *provável* que, sem o auxílio de mecanismos extra, os escritores de *software* não possam fazer muito mais. Apesar disso, é evidente que vão aparecer mais jogos de *arcade* e cenários cada vez mais imaginativos. É por isto que nos mostramos optimistas quanto à existência, no futuro, de aventuras, jogos de estratégia e de guerra, simulações de administração, etc., ainda mais complexos, no *Spectrum*. Foram vendidos tantos destes aparelhos que é inverosímil que ele seja posto de parte a favor de um qualquer computador *XYZ,* com preço equivalente, mais recente. O futuro apresenta-se brilhante. Neste momento encontramo-nos em relação ao *QL* na mesma situação em que esses escritores anteriores se encontravam face ao *Spectrum:* um aparelho novo, capacidades excitantes, uma linguagem nova e interessante. Como o processador é diferente, vai ser necessário aprender um novo conjunto de instruções antes de ser possível utilizar um novo conjunto de instruções antes de ser possível utilizar em pleno o código-máquina, e pode prever-se a existência de um curto período de ajustamento antes de serem lançados programas verdadeiramente surpreendentes.

De que tipo virão a ser estes programas? Bom, tenho a certeza de que *não* nos vai ser apresentado o programa *QL Invaders* ou o programa *QL Scramble.* Apesar de já existir uma versão absurdamente cara de *Connect Four* (Ligação 4), o *QL* é, obviamente, digno de coisas bem melhores. Pensamos que podemos contar com complexos jogos de estratégia como *Code Name MAT* (Nome de Código: MAT), escrito para o *Spectrum*. Este programa, da Mikrogen, é uma versão muito sofisticada de *Startrek,* mas desta vez o jogador é tudo o que se interpõe entre o sistema solar livre e um receado inimigo alienígena. O jogo, que mistura imagens soberbas nas sequências de acção com pormenorizadas movimentações estratégicas dos esquadrões espaciais sob o comando do jogador, joga-se com imenso gosto. Em vez de se basear em pancadaria violenta, tendente a dar vazão a sentimentos de agressão, este jogo pede uma abordagem mais cuidadosa. Pessoalmente achamos as passagens de acção bastante difíceis de dominar, e preferíamos que fosse dado mais ênfase à parte táctica. Seja como for, é este tipo de jogo que ficaria bem quando traduzido para o *QL*.

Os dois jogos de aventuras da Talent não fazem uso pleno das capacidades do *QL*. *West* foi traduzido da versão para o *Commodore 64* (parece-nos que também se encontra agora disponível uma versão para o *Spectrum*), não tendo aparentemente sido acrescentados novos locais. Apenas é utilizada uma fracção da memória disponível do *QL,* mas, apesar de tudo, a aventura é agradável de jogar. Como é sugerido pelo título, a aventura passa-se numa cidade do Oeste bravio, com o cambaleante Tumbleweed, abutres volteantes, bandidos insolentes e tudo. Ao contrário de muitas outras, esta aventura permite a ressurreição do jogador com muita facilidade e sem penalização, mas, uma vez que os parâmetros do jogo não são restaurados, quando o jogador já tiver morrido algumas vezes vai andar a tropeçar em pilhas de cadáveres.

Ao contrário de *West, Zkul,* a outra aventura da Talent para o *QL* é, tanto quanto sabemos, uma invenção novinha em folha. Esta aventura acomoda-se ao cenário tradicional das aventuras e o protagonista segue através de cavernas húmidas e frias, com riachos subterrâneos e muitos tesouros para manter o jogador divertido. Encontram-se presentes vários elementos da aventura original *Colossal Cave*. Por exemplo, se o jogador andar muito tempo às voltas com um problema, o programa pergunta-lhe se ele quer uma sugestão! No entanto, esta vai custar-lhe alguns pontos, como acontece com a ressurreição: o jogador pode ressuscitar em qualquer altura, porém, o seu corpo materializa-se noutro local, fora do conjunto de cavernas, aonde lhe vai ser necessário voltar a entrar.

Em Zkul, o texto é adequadamente florido e espraiado, mas a

utilização da memória disponível ainda esteve tristemente abaixo das possibilidades do computador. Parece perfeitamente natural que, existindo a possibilidade de utilizar dois *microdrives,* qualquer aventura utilize a primeira *cartridge* para introduzir o programa controlador e a segunda *cartridge* para introduzir dados à medida que fossem necessários. A desvantagem deste método é, evidentemente, a despesa inicial da editora de *software* em *microcartridges,* que, finalmente, vai ter de ser suportada pelo público. Isto vai resultar em jogos bastante dispendiosos, o que pode não ser demasiado negativo se os jogos em si forem bons.

Alguns dos jogos já mencionados neste livro seriam bem-vindos como conversões. *Lords of Midnight* seria uma excelente conversão do *Spectrum* para o *QL,* expandindo o mapa e o apoio visual, bem como o número de personagens. No entanto, a aventura pela qual qualquer admirador de *Sinclair* espera é *The Lord of the Rings,* da Melbourne House, que recentemente comprou os direitos da obra para jogos de computador. Esta aventura poderia ficar soberba no *QL,* com sequências de batalha, demandas, etc., todas a decorrerem ao mesmo tempo.

De momento é este o panorama das aventuras existentes para o *QL,* mas não vai permanecer assim por muito tempo.

CAPÍTULO 4

Geradores de aventuras

É necessário mencionar aqui um último tipo de programas de aventuras: o gerador de aventuras.

Há dois tipos de gerador de aventuras, sendo um deles derivado do outro. Scott Adams e a Infocom, nomes que já devem ser familiares ao leitor pois já foram mencionados em capítulos anteriores, criam os seus programas com a ajuda de um gerador principal (que deve estar segurado em muitos milhões). Este método é utilizado pela maior parte das companhias especializadas em jogos de aventuras. O cenário da aventura, incluindo locais, objectos e quebra-cabeças, é escrito sob a forma de uma ficha. O programa gerador é então passado sendo introduzidos todos os dados. E pronto! Eis outro enorme sucesso comercial.

Claro que aqui existe o problema de as aventuras resultantes poderem ser demasiado semelhantes. De facto, a maior parte dos jogos das companhias mencionadas acima são de certo modo parecidos com outros da mesma proveniência. O jogador pode geralmente distinguir um jogo da Infocom de um jogo de Scott Adams, e assim por diante. O truque utilizado por estas duas companhias e também, em menor escala, por outras, para impedir que os seus jogos se tornem estereotipados, é escrever um cenário interessante. Isto, em primeiro lugar. Seguidamente, e, no caso Infocom, tendo igual importância, deve ser utilizado para desenvolver este cenário um texto espirituoso e que tenha a capacidade de entreter o jogador. Como vimos, Scott Adams aborda o problema de forma bastante diferente da que é utilizada pela Infocom: as descrições são bastante lacónicas, mas os quebra-cabeças são bastante difíceis. A

linguagem da Infocom é muito mais floreada (ainda que isso possa ser suprimido, se o jogador quiser), e o enquadramento das cenas soberbo. Além disso, os quebra-cabeças são bastante difíceis, mas a dedução lógica e as situações novas são premiadas.

Enquanto as aventuras da Infocom consistem apenas em texto, a Adventure International, de Scott Adams, mistura, agora, nos seus jogos, texto com imagem. Os jogos originais de Scott Adams consistiam apenas em texto, mas foi escrito posteriormente um gerador de imagens que permitiu a conversão das histórias mais antigas de acordo com a nova necessidade de apoio visual. Agora, todas as aventuras de Scott Adams incluem imagens. A primeira aventura com imagens a ser lançada para aparelhos britânicos foi *The Hulk,* que provocou reacções diversas. O gerador de imagens também foi lançado comercialmente para o *Atari* com o nome de SAGE (Scott Adams' Graphic Editor), de forma que podem ser incluídas imagens nos programas pertencentes ao jogador — mas vai ser preciso esperar muito tempo até que Adams se decida a lançar o seu gerador de texto!

Entretanto, na Grã-Bretanha, apareceram no mercado vários geradores. De facto, os *games machines* estão na moda, este ano, e apresentam títulos como *Fifth* (Quinto), da CRL, *Scope* (Âmbito), da ISP, e *White Lightening* (Relâmpago Branco), da Oásis. Destinam-se principalmente a jogos de *arcade,* mas também há alguns para jogos de aventuras.

Dungeon Builder (Construtor de Masmorras) da Dream Software, é uma utilidade muito simpática que permite ao jogador compor uma aventura completa com imagens. É um programa muito fácil de utilizar, ainda que a construção das imagens possa pôr os seus problemas. O excelente manual guia o autor aprendiz através dos vários estádios. Os resultados são bastante bons, embora não haja lugar para muito texto e as imagens se delineiem com bastante lentidão quando o jogo está a correr. E não se deixem enganar pelas masmorras do título: não é absolutamente necessário que o cenário seja subterrâneo.

Todavia, entre os programas geradores de aventuras, aquele que teve mais sucesso e é mais utilizado é com certeza *The Quill* (A Pena), da Gilsoft do País de Gales. Muito tem sido escrito acerca deste programa e muito tem sido escrito *por* ele, por isso não vamos entrar em pormenores aqui. Basta dizer que o manual, ainda que não seja tão simpático quanto o de *Dungeon Builder,* é real-

mente exaustivo e deve ajudar um autor mesmo noviço a escrever uma aventura que pareça profissional. Mesmo as maiores companhias de *softaware* acham que é necessário ter muito cuidado para impedir que as aventuras escritas por intermédio de um gerador se tornem cópias umas das outras. Algumas das aventuras que, de momento, se encontram disponíveis comercialmente rodeiam esse problema melhor que outras. Se bem que alguns dos parâmetros possam ser alterados (por exemplo, é possível mudar o conjunto das personagens e, se o utilizador tiver experiência suficiente, as imagens podem ser implementadas), o autor fica, no essencial, preso à concepção de aventura da Gilsoft. É claro que isto não é mau, mas em geral um jogo gerado por *The Quill* é reconhecível. Algumas das mensagens, por exemplo *«Quit: are you sure you want to quit now?»* (Desistir: tem a certeza de que quer desistir agora?), não podem ser modificadas. Por isso, o cenário e a novidade da aventura tornam-se ainda mais importantes. Foram lançadas algumas aventuras que tiram partido de uma forma de embalagem fora do vulgar (por exemplo, incluindo um frasco de comprimidos que pode servir de pista), ainda que a maior parte delas continue a ser vendida na tradicional caixa de cassetes.

Se quiser ver uma aventura gerada por *The Quill* antes de comprar esta utilidade, que presentemente se encontra disponível para o *Spectrum* e o *Commodore 64,* a Gilsoft lançou recentemente a sua Colecção de Ouro. A primeira parte da colecção compreende seis ou sete programas, sendo uns da autoria de escritores da própria Gilsoft e outros devidos a autores externos que enviaram à editora as aventuras que tinham escrito. Aparentemente, a Gilsoft recebe todos os meses montanhas de fitas.

Foram programas como *The Quill* que nos inspiraram a escrever os programas para este livro. Em vez de mostrar ao leitor como escrever uma aventura em texto, pensámos que era possível que fosse bastante mais agradável apresentar-lhe um Gerador de Aventuras e um jogo adequado a este gerador de aventuras. Passemos pois à segunda parte de *Jogos de Aventuras para o Sinclair QL,* e comecemos a programar.

SEGUNDA PARTE

Os jogos de aventuras

CAPÍTULO 5

Como usar os programas

Recomenda-se ao leitor que leia esta secção antes de teclar seja o que for no *QL*. Quando o leitor tiver digerido tudo, será a altura de começar a premir o teclado. Como uma ordem directa, escreva AUTO: isto assegura que a listagem comece em 10 e depois aumente com incrementos de 10 de cada vez que se carregar em ENTER. Assim, não é necessário continuar a verificar os números das linhas.

«QLAG» E «QAD»

Um jogo de aventuras é uma fantasia. Sendo o autor dessa fantasia, o leitor pode escolher qualquer período da história passada ou futura, e qualquer lugar. Pense neste exercício como se fosse escrever um romance. A fantasia pode decorrer, por exemplo, numa enorme nave espacial viajando no vasto e deserto espaço interestelar, ou num barco viquingue navegando no mar do Norte. Ainda que o sentido do termo *aventura* possa ser bastante alargado, pois é aplicado a jogos que são cada vez menos parecidos com as aventuras originais, o programa básico que vai ser «composto» neste livro pertence ao tipo tradicional.

Assim, vamos permitir-nos uma fantasia e decidir que a nossa aventura vai decorrer numa masmorra húmida, fria e escura situada sob um castelo antigo. Sendo habitada por monstros de todos os tipos, é o género de lugar onde se necessita de alguns companheiros, pelo que vamos ter um grupo de três personagens em vez de uma única.

UTILIZAÇÃO DE «QLAG»

Depois de ter introduzido a listagem do capítulo 6, o jogador deve salvar (SAVE) o programa antes de o fazer passar (RUN). De facto, deve sempre salvaguardar-se um programa antes de o fazer passar.

Depois de uma inicialização de cerca de vinte segundos é apresentado ao jogador o *menu* principal. As opções disponíveis são:

LOAD DATA FROM MICRODRIVE (1)
(introduzir dados do *microdrive*)
ALTER THE MAP (2)
(alterar o mapa)
ALTER THE OBJECTS (3)
(alterar os objectos)
ALTER THE MONSTERS (4)
(Alterar os monstros)
ALTER THE CHARACTERS (5)
(Alterar as personagens)
END THE PROGRAM (6)
(finalizar o programa)

Quando o programa passa pela primeira vez é evidente que não há dados para introduzir (a primeira opção utiliza-se se se tiverem salvaguardado dados numa sessão anterior). Assim, a primeira escolha do leitor vai ser a segunda opção, «alterar o mapa». Carregando em 2 a imagem no visor desaparece, sendo o *menu* substituído por uma quadrícula numerada de 12 × 24, situada no topo esquerdo do visor. Isto é o mapa da aventura. No centro do quadrado ou «cela» inferior esquerdo vê-se um pequeno ponto branco, o cursor. Usando as teclas do cursor o jogador pode movê-lo através da quadrícula. À medida que ele entra em cada cela, vai sendo apresentada informação sobre ela no topo direito do visor.

Quando se começa, todas as celas têm a descrição «nondescript», e estão vazias. Seleccionando E (para EXITS, saídas) ou D (para DESCRIPTION, descrição) podem ser colocadas de acordo com a vontade do jogador saídas em cada cela, e as celas podem ser baptizadas. Depois de o jogador ter introduzido através do teclado

a descrição da cela, o computador pergunta se há aí algum monstro ou algum objecto: podem existir ambos e até mais que um de cada.

A partir das opções 3, 4 e 5 é possível alterar os nomes dos objectos (criando assim «*The Golden Sword*», a espada dourada, ou «*King Tony's Magic Stone*», a pedra mágica do rei Tony) e as forças e as fraquezas dos monstros e das personagens.

As modificações nas forças e fraquezas conseguem-se através do uso de «*spreadsheets*» que contêm os dados necessários para alterar os «*stats*» de personagens e monstros — ou seja, informação sobre a sua avidez (GREED), sobre os seus feitiços (SPELLS), sobre as suas capacidades de ataque (ATTACK) e defesa (DEFENCE), etc. Inicialmente, é atribuído a todos estes parâmetros um valor 0, para cada monstro ou personagem, mas movendo o cursor através da *spreadsheet* é possível alterar este valor.

A avidez controla a vontade de um monstro aceitar um presente, enquanto a fúria (ANGER) controla a sua prontidão a atacar. As personagens podem, por vezes, furtar-se a uma luta, mas pode também acontecer que sejam os monstros a fazê-lo. O presente (GIFT) é o número referente ao objecto que o monstro aceita automaticamente como suborno. Os monstros devem ter, geralmente, valores baixos de ataque e defesa no início do jogo de modo que, à medida que a força do grupo vai aumentando devido à prolongada exploração, possam ser dados aos monstros poderes cada vez maiores. O valor de combate de um monstro está relacionado com o quadrado do valor que lhe é atribuído nas *spreadsheets*, enquanto as personagens têm apenas o valor exacto que lhes foi atribuído. No entanto, as personagens têm bonificações devidas aos objectos que transportam.

Finalmente, os dados gerados desta forma podem ser arquivados em *cartridges* para serem utilizados em QAD, o nosso segundo programa, que vai usar todos estes dados para passar a aventura. Quando o jogador seleccionar a opção número 6 — END THE PROGRAM — vai ser-lhe pedido que dê um nome ao conjunto de dados (na realidade o programa salvaguarda dois processos, um para números e outro para letras). Quando fizer passar o programa principal, utilize esse nome para introduzir os seus dados.

UTILIZAÇÃO DE «QAD»

É perfeitamente possível o jogador sentar-se ao teclado com um mapa vazio à sua frente e planear um jogo a partir do nada. No entanto, planear tudo de antemão é um método melhor, especialmente se se prevê a criação de uma grande aventura. Não é necessário definir todos os números nesta altura, mas o jogador pode estabelecer um mapa e as localizações dos monstros e dos tesouros, bem como a utilidade desses tesouros. Deve prestar especial atenção às entradas e saídas. Não incluímos aqui uma rotina para a criação de armadilhas temporais ou saídas que levam a locais remotos (aqui, todas as saídas conduzem a divisões fisicamente adjacentes), mas isso pode ser simulado através do uso de objectos amaldiçoados (ver adiante) e de descrições imaginativas. A propósito de locais, aqui vão dois avisos:

1) *Não coloque* um monstro no primeiro local, porque o seu grupo ficaria encurralado!
2) *Assegure-se* de que cada saída de uma divisão tem uma entrada correspondente na divisão adjacente. É claro que o jogador pode desejar, utilizando esse método, transformar essa divisão numa armadilha de onde o grupo só possa sair no caso de possuir um objecto mágico. Porém, isto deve ser o resultado de um planeamento cuidadoso e não do acaso. Também é boa ideia estar razoavelmente seguro das descrições de locais (de modo que o trabalho de tecleio seja reduzido ao mínimo) e do objectivo do jogo.

A estrutura do nosso jogo de aventuras foi construída de modo que ele possa ser jogado de várias maneiras. O jogo simples não passa do «mata e esfola»: os jogadores vagabundeiam pelo local onde o jogo se passa, encontrando monstros, matando-os e descobrindo tesouros. A finalidade deste tipo de jogo é alcançar uma pontuação elevada, que é medida através da quantidade de tesouros acumulada, do número de monstros mortos e do número de vezes em que o jogador conseguiu sobreviver. Um tipo de jogo mais compensador acrescenta a ideia de um objectivo final, constituído por um objecto único que tem de ser encontrado.

Se o último jogo lhe agrada mais que o anterior, deve indicar um objecto como sendo o objectivo final. Isto é feito conferindo a esse objecto o valor «2», na coluna LC da *spreadsheet* de dados dos objectos. Para fazer que exista um único exemplar de um determinado objecto deve escrever «0», e para criar muitos objectos iguais a esse deve escrever «1».

Há vários outros aspectos do jogo que podem ser definidos pelo utilizador. Dado que o jogo trata primordialmente de combate, muitas das variáveis afectam-no ou são por ele afectadas de algum modo. Na secção de descrição de objectos do Gerador *(Procobdat),* o jogador pode declarar, além do tipo de local (LC), se um objecto está amaldiçoado e se é o flagelo (BANE) de algum monstro.

BANE (flagelo): O parâmetro BN contém o número do monstro para o qual o objecto é um flagelo. Assim, se as personagens tentarem combater o monstro certo com esse flagelo, o monstro entra em fuga.

CURSE (maldição): O parâmetro CS é usado para denotar um objecto amaldiçoado: 0 para um objecto não amaldiçoado, 1 para um objecto que reduz a eficácia do ataque do seu possuidor numa situação de combate, 2 para um objecto que causa ferimentos ao seu possuidor durante o combate, e 3 para um objecto com um encantamento que por vezes transporta todo o grupo para um novo local ao acaso. Os aventureiros podem descobrir que têm na sua posse um objecto amaldiçoado quer usando-o em combate (o que é um método assaz drástico) quer através da intervenção fortuita de uma voz de origem indefinida *(Procvoice)* que, por vezes, quando o grupo está a descansar, se digna fornecer-lhe informações (acerca não só de objectos amaldiçoados mas também do paradeiro de certos tesouros).

A propósito, descansar traz o benefício de incrementar ao acaso uma variável para uma personagem *(Procrest).* Contudo, também há a hipótese de ser chamada uma rotina ao acaso que, no caso de o grupo estar amaldiçoado, o pode transportar a um lugar ao acaso, criar um lago mágico (beber a água desse lago pode aumentar ou diminuir o valor de certas variáveis) ou arranjar uma surpresa sob a forma de um monstro.

COMBAT (combate): Em combate, um ataque furioso acrescenta os valores de ANGER aos de ATTACK, mas subtrai ANGER de DEFENCE. No entanto, CAUTION (cautela) usa NERVES (nervos) para suplementar DEFENCE mas reduzir ATTACK. Assim, um ataque cauteloso reduz as possibilidades, tanto do inimigo quanto do jogador, de atingirem os seus alvos respectivos, enquanto um ataque furioso tem o efeito contrário. SPELLS (feitiços) são projécteis — quase nunca falham, mas podem não desencadear o seu efeito ao atingirem o alvo e, de qualquer forma, nunca causam prejuízos maiores que três pontos.

Os monstros morrem quando o seu valor de ATTACK ou o seu valor de DEFENCE atingem o valor 0 ou valores inferiores enquanto as personagens só morrem se *ambos esses valores* descerem a 0.

Um planeamento cuidadoso permite a criação de uma aventura que misture os elementos de quebra-cabeças e de combate. A «masmorra» pode ser virtualmente um labirinto estocástico, mas, usando inteligentemente as descrições, o jogador pode torná-la mais estruturada. Por exemplo, uma divisão pode ter um abutre, e outra uma pena de abutre. Esta pena pode ser quer o flagelo quer um presente para o monstro. Para alcançar a pena, as personagens podem ter de derrotar previamente três ogros. Porém, só vão poder fazê-lo se encontrarem uma poção da força de ogro (que dá altos valores de ATTACK e DEFENCE). No entanto, é possível que a poção esteja amaldiçoada, de forma que há o risco de o grupo ser transportado a qualquer outro ponto da masmorra e desperdice a sua força.

Como se vê, o método básico consiste em tornar um objecto na chave para a ultrapassagem do próximo monstro, o qual, por sua vez, está de guarda a outro objecto que é a chave para a ultrapassagem de outro monstro, e assim sucessivamente. É capaz de ser boa ideia utilizar uma poção ou uma garrafa como um objecto geral que pode ser encontrado num local qualquer da aventura e pode ser transportado até ser necessário usá-lo. As poções e as garrafas apenas conferem as suas virtudes (quando são bebidas) à personagem que as utilizou, como acontece com todos os objectos que têm 1 como valor LC, enquanto outros objectos afectam os valores de ATTACK e DEFENCE de cada vez que há uma luta. Por outro lado, todas as outras potencialidades dos objectos únicos são passadas para o seu possuidor no momento em que este se apodera deles, de forma que os objectos únicos podem ser bastante poderosos. Um livro de feitiços, por exemplo, pode transferir para a personagem nove feitiços.

Tente dar uma certa lógica a alguns objectos. Pode, talvez, fornecer uma pista no nome do objecto. Por exemplo, um globo curativo pode conferir pontos de DEFENCE, uma espada torcida pode estar amaldiçoada, um machado gigante pode ser um presente para um ogro, uma pluma vermelha pode ser um presente para um flamingo. É possível utilizar como objectivo final algo relacionado com o monstro. Por exemplo, um bordão de feiticeiro pode estar guardado em celas próximas por um exército de *trolls,* estando o poderoso feiticeiro no centro do labirinto.

Quando tiver acabado o trabalho hercúleo de teclar as listagens (pense só no tempo necessário para as escrever!) vai achar muito gratificante estar apto a criar e jogar um número infinito de aventuras com muito poucas maçadas. Se estiver particularmente satisfeito com uma aventura criada por si com a ajuda do QLAG e quiser partilhá-la, teremos muito prazer em vê-la.

Agora, antes de calçar os seus dedos de dactilógrafo, aqui vão algumas notas que deve achar úteis.

NOTAS SOBRE O USO DE «QLAG»

1. Declarar as saídas na rotina do mapa *(Procmap)* estabelece automaticamente a saída correspondente na divisão adjacente. Para retirar a saída de uma divisão (por exemplo, para criar vias de sentido único), seleccione a opção de saída e pressione uma tecla qualquer que não seja N, S, E ou W. No entanto, retirar saídas apenas produz esse efeito na divisão em questão e *não* na divisão que lhe fica adjacente. Isto pode levar à criação de uma divisão onde é possível entrar mas de onde é impossível sair. Assim, se quiser remover completamente uma saída, tenha o cuidado de a retirar de ambas as celas.

2. Para terminar a rotina de descrição de objectos *(Procobject)* é necessário escolher um objecto e, seguidamente, premir E (de END) para finalizar. Não é possível parar a rotina sem requerer um objecto.

3. Se quiser modificar parte da descrição de uma divisão *(Procdescribe)* tem de se repetir toda a rotina. Também não é possível mudar os monstros que se encontram numa cela sem voltar a teclar a descrição.

4. Visto isto, é essencial um *bom planeamento:* antes de começar, faça um mapa do conjunto das masmorras, decida o que quer colocar em cada cela, e mantenha-se fiel ao seu plano. Alguns momentos gastos nesta altura poupam muito tempo na fase de *input* dos dados.

5. Na rotina dos monstros *(Procmonster),* ATTACK e DEFENCE são usados em combate, ANGER decide se um monstro vai atacar ou não, SPELLS são projécteis mágicos, GREED determina se um monstro quer ou não quer um tesouro e NERVE é uma medida de cobardia/valentia.

6. O valor atribuído a cada objecto *(Procobadt)* é um modificador aplicado à personagem que o transporta no que diz respeito a ATTACK, DEFENCE e SPELLS.

Assim, *CS* é maldição (CURSE). Uma maldição do tipo 1 reduz as possibilidades de atingir um oponente em combate, enquanto o tipo 2 leva a personagem a atingir um amigo em vez de um inimigo. Finalmente, o tipo 3 origina o transporte do grupo para um local ao acaso.

LC especifica as propriedades de localização de um objecto. Assim, 0 aplica-se a um objecto único, que aparece apenas uma vez na masmorra. O valor 1 é atribuído a um objecto de que podem existir vários exemplares, correspondendo a todos eles a mesma descrição. O valor 2 é aplicado ao objecto que é procurado por todas as personagens. Quando esse objecto for encontrado, ganha-se o jogo.

BN é aplicado a um objecto que seja o flagelo (BANE) do monstro que se encontra num determinado local. Trata-se, obviamente, de um objecto bastante poderoso, pois, se uma das personagens do grupo o transportar, o monstro é automaticamente vencido. No entanto, só os monstros designados pelos números de 1 a 9 são afectados desta forma, o que torna os outros monstros potencialmente mais fortes. Um valor BN = 0 significa que o objecto não afecta nenhum monstro. Se forem introduzidos valores maiores que os permitidos, são registados por QLAG, mas não vão ter qualquer efeito no jogo, ou seja, contam como se fossem 0.

7. As variáveis respeitantes às personagens são aproximadamente as mesmas que as respeitantes aos monstros.

SAÍDAS

Esta secção pode, de momento, não ter grande significado para o leitor, mas à medida que for teclando o programa vai-se tornando mais clara. Não entre em pânico se, de momento, não a conseguir compreender.

Para cada divisão há quatro possibilidades de saída: Norte (N), Sul (S), Leste (E) e Oeste (W). Em QLAG é necessário ser capaz de assinalar estas saídas num mapa e em QAD é preciso desenhar as portas apropriadas, com excepção da porta pela qual se entra na divisão.

Isto requer, infelizmente, alguma matemática elementar.

IDENTIFICAÇÃO DAS SAÍDAS

Para identificar as saídas poderíamos usar uma disposição em três dimensões: «número de celas horizontalmente», «número de celas verticalmente» e «número de saídas possíveis». Isto daria uma disposição de $12 \times 24 \times 4 = 1152$ para cada uma das quatro saídas possíveis. No entanto, isto representa um grande desperdício, se tivermos em conta que a informação que queremos é binária: uma determinada saída ou está disponível ou não está (é como se estivesse ligada ou desligada). Poderíamos poupar um quarto deste es-

paço se usássemos apenas um *bit* para representar cada saída possível, o que é a última dimensão da nossa disposição.

Suponhamos que usamos o *bit* 0 (o que se encontra mais à direita) para S, o *bit* 1 para W, o *bit* 2 para N e o *bit* 3 para E. Isto significa que o Sul é representado pelo número decimal 1, o Oeste por 2, o Norte por 4 e o Leste por 8. Vamos chamar a estes números decimais os números das saídas e aos números dos *bits* (0-3) os códigos das saídas. As nossas saídas teriam a seguinte aparência:

```
            N(2)
W(1)                E(3)
            S(0)
```

Se numa divisão existirem as quatro saídas, a disposição conterá o número 15 = 1 + 2 + 4 + 8. Isto significa que o valor decimal das saídas da cela (a que vamos chamar o código da cela) é 15.

Nos programas, atribuímos os valores do código de saída usando a cadeia «SWNE» e subtraindo 1 às posições das saídas nessa cadeia. S é o primeiro elemento da cadeia, e 1—1 = 0, que é o código de saída de S. W é o segundo elemento e 2—1 = 1, que é o código de saída para W, e assim sucessivamente.

Se quisermos encontrar o código de uma cela usamos a fórmula básica:

código da cela (valor decimal das saídas da cela) = soma de todos os números de saída, sendo cada número de saída 2^ ((posição na cadeia «SWNE») − 1)

Assim, quando numa divisão estão presentes as quatro saídas, o valor decimal das saídas da cela vai ser:

2^(1 − 1) + 2^(− 1) + 2^(3 − 1) + 2^(4 − 1)
= 2^0 + 2^1 + 2^2 + 2^3
= 1 + 2 + 4 + 8 = 15

Lógica «bitwise»

Para podermos usar esta informação no QLAG e no QAD precisamos de usar «lógica *bitwise*».

Se teclarmos

PRINT 4 && 12

obtemos como resultado «4». Isto acontece porque o operador && leva à comparação de dois *bytes* de tal forma que se o mesmo *bit* estiver activado em *ambos* os *bytes*, permanece activado, mas se um *bit* estiver activado apenas em *um* dos *bytes*, esse *bit* será reposto a zero.

Uma vez que o número decimal 12 acciona os *bits* 2 e 3 mas o número decimal 4 apenas acciona o *bit* 2, o único *bit* comum aos dois números é o *bit* 2. Em notação decimal, o *bit* 2 é 4, por isso, a função *bitwise* AND dá-nos o valor 4. Se compararmos os números decimais 12 e 1, em que o número decimal 12 acciona os *bits* 2 e 3 e o número decimal 1 acciona o *bit* 0, a função *bitwise* dá-nos o valor 0.

O *QL* trata 0 como sendo «falso» e qualquer outro número como sendo «verdadeiro», e age dessa forma numa condição do tipo

IF 4 && 12 THEN PRINT «IT WORKS!!»

Por outras palavras, se compararmos qualquer número sucessivamente com 1, 2, 4 e 8 obtemos 0 se o *bit* correspondente não estiver accionado, e 1, 2, 4 ou 8 se esse *bit* estiver accionado.

A marcação de portas em «QLAG»

Se, na quadrícula do mapa, colocarmos o cursor no centro de uma cela para marcar uma dada saída, ele vai ter de se deslocar uma distância que pode ser positiva ou negativa (multiplicada por + 1 ou por —1) ao longo de um dos dois eixos, vertical ou horizontal. Por exemplo, para marcar a saída Norte o cursor tem de se mover positivamente no eixo vertical, e para marcar a saída Oeste tem de se mover negativamente no eixo horizontal.

Usando a função *bitwise* AND podemos traduzir as direcções que são introduzidas em *Procentrance* nos códigos de saída 0-3. Vamos agora ter de traduzir essas direcções em + 1 ou − 1 (o sentido no qual se dá a movimentação do cursor sobre o eixo) e também determinar a qual dos eixos vai essa deslocação ser aplicada. Se o código da saída (referido na rotina como «wall») for um número par (0 ou 2), trata-se do eixo vertical (S ou N); se for ímpar (1

ou 3), então trata-se do eixo horizontal (W ou E). Para ver se o número é par divide-se por 2, usando tanto a divisão real como a divisão inteira (/ e DIV). Se o resultado das duas divisões coincidir, o número é par; caso contrário é ímpar.

O movimento de cada eixo está contido em *Xw* para horizontal e em *Yw* para vertical. No caso de um número par, como o movimento é vertical, *Xw* é 0 e *Yw* tem de ser +1 ou −1. Para descobrir de qual destas duas hipópteses se trata subtraímos ao nosso número par (0 ou 2) o número 1. Se o número for ímpar, então *Yw* vai ser 0 e *Xw* vai ser +1 ou −1 (este é o caso de ELSE no procedimento *Procdoor_draw* de QLAG). Para saber se *Xw* é positivo ou negativo subtraímos ao nosso número ímpar (1 ou 3) o número 2.

As coordenadas *Xw* e *Yw* são, então, simplesmente acrescentadas às coordenadas do cursor preexistentes, e o cursor desloca-se para a posição requerida, de modo a marcar a porta no mapa.

Desenhar portas em QAD

Desenhar portas em QAD não é um problema tão fácil de resolver como o anterior. É necessário não só interpretar o código de saída como desenhar apenas as saídas que o jogador pode ver (ou seja, excluindo aquela por onde o grupo entra), e colocá-las correctamente orientadas. Isto quer dizer que precisamos de saber por qual das saídas foi feita a entrada, sendo o seu valor guardado na variável «*entrance%*».

Para desenhar as portas que o jogador pode ver de forma que as saídas estejam correctamente orientadas (ou seja, ficando a porta de Leste à esquerda da porta Norte, etc.), podemos atribuir códigos de porta à «porta da direita», à «porta da esquerda» e à «porta em frente».

Se o jogador se deslocar para Norte, entra na divisão seguinte pela porta Sul (vejamos: está virado para Norte, por isso a porta à sua frente vai ser a porta Norte, e entrou pela porta Sul). Usando os códigos de saída da nossa cadeia «SWNE»-1, a porta por onde entrámos, S, tem como código 0; W fica à esquerda e tem código 1, um a mais que zero; N fica em frente, portanto o código dessa porta é 2, dois a mais que zero; E fica à direita, por isso o código dessa porta é 3, três a mais que zero.

Se entrarmos pela porta Oeste, a porta de entrada teria código 1, que é o código da saída W; a porta da esquerda seria N=2, um a mais que 1, o valor da entrada; a porta em frente, seria E=3, dois a mais que 1. Até aqui parece que podemos atribuir às portas os códigos de 1 para «porta à esquerda» e 2 para a «porta em frente». Mas neste caso a porta à direita é S=0, o que é um a menos que 1 e não três a mais, como seria de esperar.

No entanto, se tratarmos da divisão como um círculo, e andarmos às voltas no sentido dos ponteiros do relógio, podemos numerar as saídas de 0 até 3 e continuar subsequentemente a atribuir os números 4, 5, etc. É claro que, como só temos quatro saídas, a porta 4 e a porta 0 são a mesma (bem como as portas 8, 12, etc.). Assim, para o efeito de estabelecer códigos de porta, podemos tratar a porta 0 como se fosse a porta 4, o que dá três a mais que 1, que é o que nós queríamos.

Assim, podemos dizer que o código para «porta à esquerda» é 1, para «porta em frente» é 2, e para «porta à direita» é 3. Agora dispomos de códigos para as portas (à esquerda, em frente e à direita, 1-3) e para as saídas (0-3); falta-nos um código para a entrada.

Precisamos de transformar o valor da porta de saída de uma divisão no valor da porta de entrada da divisão seguinte. Como já vimos, se se sair de uma divisão pela porta Norte, entra-se na seguinte porta Sul — ou seja, entra-se pela porta «em frente». Se nos lembrarmos de que o código da «porta em frente» é 2, vemos que temos necessidade de uma cadeia de códigos de entrada que seja inversa da dos códigos de saída — esta cadeia de códigos de entrada é «NESW»-1. Assim, se deixarmos uma divisão pela porta Norte, pomos o valor do seu código de entrada (0) na variável *entrance%*, o que significa que, quando entramos na divisão seguinte, *entrance%* contém os valores do código de saída para Sul (que é 0, como se vê a partir da cadeia «SWNE»-1).

Desta forma, na rotina *Procgraphic* de QAD, pegamos no nosso código de cela (contido em «*map%*») e fazemos uma comparação *bitwise* entre ele e 2^0, 2^1, 2^2 e 2^3 (um de cada vez, claro, por meio de um ciclo). Se o teste der «verdadeiro», o passo seguinte é ver se este era o valor da porta por onde entrámos; caso contrário é chamada a rotina *Procdoor_draw* (para desenhar as portas), tendo um dos números de 0 até 3 como parâmetro. Este processo é repetido até terem sido verificadas todas as saídas possíveis.

Para efeitos de programação, podemos pensar nos códigos de porta como sendo o resto do cálculo:

Código de porta = (código de saída + 4 − código de entrada) MOD4

(ver *Procdoor_draw*). Assim, se entrar numa divisão através da porta Leste (a última deslocação que fez foi para Oeste), e se quiser saber onde vai desenhar a saída Sul, o procedimento é:

1) O código de entrada de W é 3 (posição em «NESW» − 1).
2) O código de saída para S é 0 (posição em «SWNE» − 1).
3) 0+4=4.
4) 4-Código de entrada (3)=1.
5) O resto da divisão de 1 por 4 é 1.
6) 1 é o código para «porta à esquerda».
7) Assim, a saída S deve ser desenhada na parede da esquerda (ou seja, à esquerda da porta S).

PRINCIPAIS VARIÁVEIS EM «QLAG»

map% — divisões na horizontal, divisões na vertical, código de saída.
número de objecto
monstro
número de monstros

map$ — divisões na horizontal, divisões na vertical, comprimento máximo de cadeia

object$ — número de objectos, pré-descrição
nome de objecto
pós-descrição

object% — objectos, ataque
defesa
feitiço
maldição
local
flagelo

monster% — monstros, ataque
defesa
feitiço
fúria
avidez
presente

character% — os mesmos parâmetros que *monster%, exceptuando* presente

CAPÍTULO 6

O gerador de aventuras: QLAG

Este capítulo e o seguinte contêm as listagens para os programas QLAG e QAD, tendo sido incluídos c omentários onde isso foi achado necessário. Tanto quanto possível, as rotinas são discutidas pela ordem por que são chamadas pelo programa, de modo que o leitor possa seguir o sentido dos programas. No entanto, nenhum dos programas terá qualquer actuação até ter sido introduzido o programa completo.

Ao longo das discussões os dois programas, aconselhamos por vezes a consulta do manual do *QL*. Como acontece com o sistema operador do computador, o manual está permanentemente a ser actualizado. Neste livro, referimo-nos à versão de 6/84.

CONTROLADOR DO PROGRAMA

```
100 REMark QL ADVENTURE GENERATOR
1984 N RICHARD WILLIAMS & TONY BRIDGE
110 RESTORE
120 init
130 REPeat supervisor
140 menu
150 get "1","6"
160 g=g$
170 SELect ON g
180 ON g = 1
```

```
190 infile
200 ON g=2
210 CLS
220 map
230 ON g=3
240 CLS
250 object
260 ON g=4
270 CLS
280 monster
290 ON g=5
300 CLS
310 character
320 ON g=6
330 EXIT supervisor
340 END SELect
350 END REPeat supervisor
360 file_out
370 STOP
```

Comentário sobre o controlador do programa

Este é o controlador principal para todo o programa, e chama cada rotina principal à medida que é necessário. A primeira rotina chamada é:

Procinit

```
12000 DEFine PROCedure init
12010 CLS
12020 AT 10,10:PRINT "Initialising...
."
12030 DIM map%(12,24,4)
12040 DIM map$(12,24,60)
12050 DIM object$(36,3,12)
12060 DIM object%(36,6)
12070 DIM monster$(16,8)
12080 DIM monster%(16,6)
```



```
12090 DIM character%(6,5)
12100 DIM character$(6,8)
12110 DIM category$(9,4)
12120 DIM axis$(6,2)
12130 FOR i = 1 TO 36
12140 READ object$(i,2)
12150 FOR j = 1 TO 6
12160 READ object%(i,j)
12170 NEXT j
12180 NEXT i
12190 FOR i=1 TO 12
12200 FOR j=1 TO 24
12210 FOR k= 1 TO 4
12240 map$(i,j)="nondescript"
12250 NEXT k:NEXT j:NEXT i
12255 monsterskilled=0:turns=0:h=1:v=
1
12256 char$="123"
12260 compass$="SWNE"
12270 FOR i=1 TO 16
12280 READ monster$(i)
12290 NEXT i
12300 FOR i=1 TO 9
12310 READ category$(i,1 TO 4)
12320 NEXT i
12330 FOR i = 1 TO 6
12340 READ character$(i)
12350 FOR j= 1 TO 5
12360 character%(i,j)=RND(1 TO 9)
12370 NEXT j
12380 NEXT i
12390 FOR i=1 TO 6
12400 READ axis$(i)
12410 NEXT i
12420 END DEFine init
```

Comentário sobre «Procinit»

A rotina dimensiona várias cadeias, cada uma das quais vai conter informação sobre um determinado aspecto do mapa em cada ponto. O mapa vai tomar a forma de uma matriz de 12×24. Cada uma das celas desta matriz vai poder (ou não) conter um determinado número de monstros e de objectos. O mapa vai ser descrito por uma frase curta ditada pelo utilizador (ver *Procdescribe*). Na linha 12260, *compass$* contém a cadeia de direcções de saída que foi discutida no capítulo anterior.

Map% é uma variável inteira que contém, no seu terceiro elemento, informação sobre as saídas de cada cela — existindo quatro direcções possíveis para uma saída. *Map$* contém, no seu terceiro elemento, mensagens em texto que vão ser usadas mais tarde para serem apresentadas no visor de alteração do mapa. As variáveis inteiras *object%, monster%* e *character%* contêm os valores dados aos objectos, etc., e *object$,* etc., contêm os nomes e as descrições que vão ser discutidos mais tarde. As declarações de dimensão mostram que temos 36 objectos, 16 monstros e 6 personagens. Estes encontram-se nas declarações de dados, que são as linhas a partir de 12430, sendo os dados agora lidos para as cadeias e quadros relevantes. Os dados para *category$* estão na linha 12810 e os dados para *axis$* na linha 12830. As linhas de dados são fornecidas no fim deste capítulo.

Linhas 12190-12250: Quando começamos, todas as celas (ou divisões) do mapa devem estar vazias — estes ciclos asseguram que assim seja. A linha 12240 coloca em *map$* uma mensagem com esta finalidade, que vai ser impressa mais tarde, quando for necessário.

PROCMENU

```
500 DEFine PROCedure menu
510 REMark OPEN A FULL SCREEN
520 OPEN #1, con_482x256a0X0
530 PAPER 2
540 CLS
550 AT 2,2:PRINT "Press the correct n
umber to:"
560 STRIP 0
```



```
570 AT 4,8:PRINT "load data from micr
odrive(1)"
580 AT 6,8:PRINT "alter the map
     (2)"
590 AT 8,8:PRINT "alter objects
     (3)"
600 AT 10,8:PRINT "alter the monsters
     (4)"
610 AT 12,8:PRINT "alter the characte
rs        (5)"
620 AT 14,8:PRINT "end the program
     (6)"
630 STRIP 2
640 END DEFine menu
```

Comentário sobre «Procmenu»

Agora que o controlo voltou de *Procinit,* o controlador principal do programa entra num ciclo na linha 130 (o ciclo supervisor). A inicialização já está acabada, e agora vamos regressar sempre ao menu.

Linha 520: Abre uma janela que de facto ocupa a totalidade do visor (consultar o manual, *Beginners' Guide*). A *linha 530* escolhe uma cor para a janela — aqui é o vermelho, mas é claro que o leitor pode experimentar utilizar as suas cores preferidas. A *linha 560* escolhe uma cor de *Strip* (ver o manual, *Keywords*) para imprimir as opções que se seguem. Na *linha 520,* deve notar-se que *tanto* o «X» maiúsculo *como* o «x» minúsculo são válidos.

Antes de examinar os elementos do menu, precisamos de ver a próxima rotina a ser chamada pelo controlador.

PROCGET

```
5370 DEFine PROCedure get(low$,hi$)
5380 REPeat kget
5390 g$=INKEY$
5400 IF g$>=low$ AND g$<=hi$ THEN EXI
```

```
T kget
5410 END REPeat kget
5420 END DEFine get
```

Comentário sobre «Procget»

Esta rotina vai ser chamada várias vezes ao longo do programa gerador: de cada vez que um menu é impresso, ela determina limites inferiores e superiores para o *input* numérico, dependentes dos valores em *low$* e *his$* no momento em que *Procget* é chamada. *Procget* detecta os erros de dactilografia, repetindo o processo até que seja premida uma tecla correcta (ou seja, se for feito um erro, a pressão na tecla correspondente é ignorada). No controlador (linha 150), *Procget* estabelece um limite inferior de 1 e um limite superior de 6: as pressões em teclas dentro deste limite são consideradas e as outras são ignoradas.

A primeira selecção do menu que vamos discutir é *«alter the map»* (alterar o mapa), que escolhemos premindo a tecla 2. Isto significa (linhas 200-220, no controlador) que a nossa primeira tarefa vai ser desenhar o mapa.

PROCMAP

```
1000 DEFine PROCedure map
1010 xpos=13.5:ypos=35
1020 CLS
1030 REMark CREATE THREE WINDOWS
1040 OPEN #10,scr_320x160a32x16
1050 PAPER #10,3
1060 OPEN #11,con_440x64a32x176
1070 PAPER #11,5
1080 OPEN #13,scr_120x160a352x16
1090 PAPER #13,0
1100 CSIZE #10,0,0
1110 CLS #10
1120 CLS #11
```



```
1130 CLS #13
1140 INK #10,0
1150 SCALE #10,160,0,0
1160 FOR i = 10 TO 130 STEP 10
1170 LINE #10, 10,i+20 TO 226,i+20
1180 NEXT i
1190 FOR i=9 TO 225 STEP 9
1200 LINE #10, i,30 TO i,150
1210 NEXT i
1220 INK #10,7
1230 FOR i = 97 TO 120
1240 AT #10,0,i-96 :PRINT #10,CHR$(i)
1250 NEXT i
1260 FOR i = 1 TO 12
1270 AT #10,i,0 :PRINT #10,CHR$(109-i
)
1280 NEXT i
1290 CSIZE #10,3,1
1300 PRINT #10, "     ADVENTURE MAP"
1310 FOR i = 1 TO 12
1320 FOR j= 1 TO 24
1330 IF map%(i,j,1)=0 THEN GO TO 1370
1340 FOR k=0 TO 3
1350 IF (map%(i,j,1) && 2^k) THEN doo
r_draw k,j,i
1360 NEXT k
1370 NEXT j
1380 NEXT i
1390 INK #11,0
1400 scr_mes
1410 vp=1:hp=1
1420 REMark THE CURSOR ROUTINE
1430 cur 0,0,9,10,220.5,13.5,145,35,-
2
1440 room
1450 REPeat control
1460 g$=INKEY$
1470 g=CODE(g$)
1480 SELect ON g
```

```
1490 ON g= 192
1500 cur -9,0,9,10,220.5,13.5,145,35,
-2
1510 room
1520 ON g= 200
1530 cur 9,0,9,10,220.5,13.5,145,35,-
2
1540 room
1550 ON g= 208
1560 cur 0,10,9,10,220.5,13.5,145,35,
-2
1570 room
1580 ON g=216
1590 cur 0,-10,9,10,220.5,13.5,145,35
,-2
1600 room
1610 ON g= 232
1620 CLOSE#10
1630 CLOSE#11
1640 CLOSE#13
1650 EXIT control
1660 ON g= 69,101
1670 entrance
1680 scr_mes
1690 ON g= 68,100
1700 describe
1710 scr_mes
1720 END SELect
1730 END REPeat control
1740 END DEFine map
```

Comentário sobre «Procmap»

xpos e *ypos* são as coordenadas da posição em que o cursor vai ser finalmente colocado quando a quadrícula for desenhada. O aparecimento precoce, neste comentário, destas duas variáveis tem como objectivo vermo-nos livres desse assunto!

Linhas 1040-1100: As janelas são definidas e abertas e a cor do papel é determinada. A janela # 10 contém o nosso mapa, a # 11 contém um pequeno painel em que aparece informação sobre a cela de que se está a tratar (a informação contida em *map$*). Finalmente, a janela # 13 destina-se a mensagens do sistema de indicações para o utilizador.

Linhas 1110-1140: É derminado o tamanho dos caracteres (consultar o manual, rubrica *Keywords*), o conteúdo das janelas é eliminado, e a cor da tinta é determinada.

Linhas 1150-1210: É definida a escala do eixo do *x* (ver no manual a secção *Keywords*) e o mapa é desenhado.

Linhas 1220-1280: Agora as letras (*linhas 1230-1240*) são impressas em caracteres minúsculos ao longo dos eixos do *x* e do *y* da quadrícula do mapa.

Linhas 1290-1300: Usando o maior tamanho de caracteres possível (ver no manual a secção *Keywords*), é impresso «*ADVENTURE MAP*» sob o mapa.

Linhas 1310-1380: Estas linhas desenham as saídas em cada cela do mapa, no caso de elas terem sido previamente definidas pelo utilizador. Claro que, por enquanto, esta parte do procedimento não vai encontrar saídas nenhumas e, assim, repete-se até que todas as celas tenham sido verificadas. Mais tarde, quando as saídas tiverem sido definidas e subsequentemente encontradas, *Procdoor_draw* vai ser chamada pela *linha 1350,* que utiliza um bom pedaço de matemática para descobrir se foi atribuída uma saída à cela que está a ser investigada nesse momento. Para uma explicação mais profunda deste assunto, ver a discussão sobre as saídas no capítulo anterior. O leitor deve lembrar-se de que, se remover uma saída de uma divisão, também a deve remover da divisão adjacente.

Linha 1390: é determinado que a tinta da janela # 11 seja preta e, na *linha 1400,* é chamada *Procscr_mes,* que imprime as instruções sob o mapa (na janela # 11, como vamos ver).

Linha 1410: determina a posição inicial do cursor: os valores de *vp* e *hp* (posição vertical e posição horizontal) vão ser usadas em *Proccur,* que é chamada na *linha 1430* (linha essa que também estabelece os parâmetros iniciais em *Proccur*).

Linha 1440: Esta linha chamada *Procroom.* Como vamos ver, esta rotina dá informação, na janela # 13, acerca dos monstros e objectos que foram colocados na cela que está a ser tratada, bem como a descrição que foi atribuída ao local.

Linha 1450-1730: Estas linhas constituem o ciclo de controlo. É lido o código da tecla do cursor que foi premida. Ver, por exemplo, a *linha 1490:* o número decimal 192 é o código de carácter para mover o cursor para a esquerda. Se esta tecla for premida, o primeiro parâmetro da variável *cur* passa de 0 (determinado na *linha 1430*) a -9, movendo o cursor para a esquerda. Passe a *Proccur,* linha 1750, e vai ver que o primeiro parâmetro é *xinc* — foi este que foi mudado. Porém, na *linha 1590,* o segundo parâmetro, *yinc,* é posto em -10, enviando o cursor para baixo. Na *linha 1610,* 232 é o código decimal para a tecla F(unção) 1, o que termina toda a sequência e nos envia para o menu (ou seja, voltamos para o ciclo supervisor no início do programa). Depois de cada uma das mudanças do cursor, é chamada *Procroom* para voltar a desenhar o mapa. Depois de F1 ser de novo premida, as janelas são fechadas e todos os processos encerrados.

Linha 1660: No entanto, se a tecla premida for E (de *exit,* saída), é chamada *Procentrance.* Note-se que tanto a maiúscula (*E,* código 69) quanto a minúscula (*e,* código 101) são consideradas. O utilizador não precisa de se preocupar com o facto de ser ou não necessário usar letras maiúsculas.

Linha 1680: As indicações para o utilizador, que são apresentadas no visor principal, aparecem uma vez mais (*Procscr_mes*).

Linha 1690: Se for premido D (de descrição), é chamada *Procdescribe.*

E assim continuamos, até termos visitado todas as celas de que queremos tratar. Todos os parâmetros que alterámos ao usar a rotina foram colocados no elemento final de *map%,* sendo a posição de cada cela lida para *vp* e *hp.* Assim, quando o cursor é posicionado no mapa em, por exemplo, «ft», *vp* toma o valor de f e *hp* toma o valor t. Então, o programa examina o elemento final de *map%* e encontra aí o conteúdo e as saídas da cela ft.

Vamos agora examinar com mais atenção as várias rotinas chamadas por *Procmap.*

PROCCUR

```
1750 DEFine PROCedure cur (xinc,yinc,
xint,yint,xr,xl,yu,yd,var)
1760 INK #10,3
1770 LINE #10,xpos,ypos-1 TO xpos,ypo
s+1
1780 xpos=xpos+xinc:ypos=ypos+yinc
1790 IF xpos>xr THEN xpos=xr
1800 IF xpos<xl THEN xpos=xl
1810 IF ypos<yd THEN ypos=yd
1820 IF ypos>yu THEN ypos=yu
1830 vp=(ypos DIV yint)+var:hp=xpos D
IV xint
1840 CURSOR #10,xpos,ypos
1850 REMark CURSOR POSITION
1860 CSIZE #10,0,0
1870 INK #10,7
1880 LINE #10,xpos,ypos TO xpos,ypos
1890 END DEFine cur
```

Comentário sobre «Proccur»

Esta rotina estabelece a posição do pequeno cursor em cada cela, à medida que as teclas do cursor são premidas.

Linha 1750: Os parâmetros que estão entre parênteses recebem informação que lhes é transmitida através da variável *control* (linhas 1450-1650, em *Procmap*).

Linha 1770: Desenha uma linha extremamente curta, constituída apenas por um *pixel:* o cursor transforma-se num ponto.

Linhas 1780-1820: Colocam o cursor na cela inferior esquerda do mapa (a,a) e actualizam a posição do cursor usando os primeiros parâmetros da *linha 1750.* As *linhas 1790-1820* asseguram que o cursor seja mantido dentro dos limites do mapa.

Linha 1830: Confronta a posição do cursor com a variável *map%,* permitindo ao programa saber quais os conteúdos a imprimir na janela # 10.

Linha 1840; Cobre o cursor antigo com a cor do papel, fazendo-o desaparecer. A *linha 1880* imprime o cursor na nova posição.

PROCSCR_MES

```
2830 DEFine PROCedure scr_mes
2840 CLS#11
2850 PRINT #11,"Cursor keys move curs
or, E selects exits, D sets descripti
ons, F1 to end"
2860 END DEFine scr_mes
```

Comentário sobre «Procscr_mes»

Esta rotina apaga o que se encontrava na janela # 11 e imprime o menu de opções.

PROCROOM

```
2750 DEFine PROCedure room
2760 CLS#13
2770 PRINT #13;"Cell ";CHR$(vp+96);",
";CHR$(hp+96)
2780 IF map$(vp,hp,1)<>"" THEN PRINT
#13;map$(vp,hp,1 TO 60)
2781 FOR k=0 TO 3
2782 IF (map%(vp,hp,1)&& (2^k)) THEN
PRINT #13,compass$(k+1);" ";
2783 NEXT k
2784 PRINT #13," "
2790 IF map%(vp,hp,2)<>0 THEN PRINT #
13; object$(map%(vp,hp,2))
2800 IF map%(vp,hp,3)<>0 THEN PRINT #
13;map%(vp,hp,4);" ";monster$(map%(vp
,hp,3));
2810 IF map%(vp,hp,4)>1 THEN PRINT #1
3;"s"
2820 END DEFine room
```


Comentário sobre «Procroom»

Esta rotina apaga o que se encontrava na janela # 13, que está reservada às descrições da divisão (ou cela) que estamos a inspeccionar, examinando seguidamente as variáveis *map$* e *map%*. A informação, dependente dos resultados, é então impressa na janela. Note-se, na *linha 2782,* o uso da função *bitwise* AND, que verifica as saídas existentes.

PROCENTRANCE

```
2100 DEFine PROCedure entrance
2110 CLS #11
2120 PRINT #11;"Type in all the direc
tions"!"then press enter"
2130 INPUT #11, direction$
2135 map%(vp,hp,1)=0
2140 FOR i = 1 TO LEN(direction$)
2150 d$=direction$(i)
2160 z=d$ INSTR compass$
2170 IF z<>0 THEN exproc
2180 NEXT i
2190 END DEFine entrance
```

Comentário sobre «Procentrance»

Se o jogador escolheu a opção *exit* do menu de *Procscr_mes,* esta rotina apaga o menu da janela # 11 e imprime uma nova mensagem. Quando uma ou várias direcções são tecladas (as letras N, S, E, W são suficientes) são colocadas em *direction$*. A variável *compass$* é verificada e o valor de cada direcção escolhida vai ser atribuído à variável *z (linhas 2140-2150)*. Estes valores são S=1, W=2, N=3 e E=4 (ver a linha 12260, em *Procinit,* e ainda a discussão sobre saídas do capítulo anterior). O valor de *z* é colocado em *map%*. A *linha 2170* chama a rotina seguinte.

```
11250 DEFine PROCedure exproc
11260 IF z=1 AND vp=1 THEN RETurn
11270 IF z=2 AND hp=1 THEN RETurn
11280 IF z=3 AND vp=12 THEN RETurn
11290 IF z=4 AND hp=24 THEN RETurn
11300 SELect ON z
11310 ON z=1
11320 map%(vp-1,hp,1)=map%(vp-1,hp,1)
||4
11330 ON z=2
11340 map%(vp,hp-1,1)=map%(vp,hp-1,1)
||8
11350 ON z=3
11360 map%(vp+1,hp,1)=map%(vp+1,hp,1)
||1
11370 ON z=4
11380 map%(vp,hp+1,1)=map%(vp,hp+1,1)
||2
11390 END SELect
11400 map%(vp,hp,1)=map%(vp,hp,1)||(2
^(z-1))
11410 door_draw z-1,hp,vp
11420 END DEFine exproc
```

Comentário sobre «Procexproc»

Esta rotina define as saídas. À medida que o utilizador tecla a escolha de saídas, a informação é colocada na variável *map %* e os parâmetros são passados para *Procdoor-draw*. As *linhas 11320, 11340, 11360* e *11380* usam a função *bitwise* OR (||), que acrescenta o número ao símbolo em *map %* (a não ser nos casos em que o *bit* relevante já esteja activado). Convém notar que estas linhas marcam as saídas correspondentes nas celas adjacentes. A *linha 11400* trata das saídas na divisão de que se está a tratar.

As *linhas 11260-11290* impedem que sejam desenhadas saídas nos limites do mapa.


```
2200 DEFine PROCedure door_draw(wall,
cem,brik)
2210 LOCal c:LOCal b:LOCal w
2220 c=cem:b=brik
2230 w=wall
2240 IF w DIV 2=w/2 THEN
2250 w=w-1
2260 Yw=w
2270 Xw=0
2280 ELSE
2290 w=w-2
2300 Xw=w
2310 Yw=0
2320 END IF
2330 LINE #10,(4.5+(c*9)+(Xw*4.5)),(2
5+(b*10)+(Yw*5))
2340 LINE_R #10 TO 1,1
2350 END DEFine door_draw
```

Comentário sobre «Procdoor_draw»

Esta rotina traduz os valores de direcção em movimentos sobre os eixos horizontal e vertical. A *linha 2240* verifica se o valor é par. Se esta condição se verificar, o movimento dá-se no eixo horizontal (Yw = 0 e Xw = +1 ou -1); caso contrário, a declaração ELSE, na *linha 2280,* entra em acção e o movimento dá-se no eixo vertical (XW = 0 e YW = +1 ou -1). Para mais pormenores sobre este assunto, ver a secção dedicada às saídas no capítulo anterior. As *linhas 2330-2340* contêm os cálculos para deslocar o cursor no sentido positivo ou negativo sobre os eixos horizontal e vertical e desenhar a saída.

```
1900 DEFine PROCedure describe
1910 CLS #11
1920 PRINT #11;"TYPE IN THE CELL DESC
RIPTION (max 60 characters)"
1930 INPUT #11;map$(vp,hp)
1940 CLS#11
1950 PRINT #11; "Any monster here?(Y/
N)"
1960 REPeat get_loop
1970 g$=INKEY$
1980 IF g$<>"" THEN EXIT get_loop
1990 END REPeat get_loop
2000 IF g$="y" OR g$="Y" THEN assign_
mon
2010 CLS #11
2020 PRINT #11; "Any object here? (Y/
N)"
2030 REPeat get_loop
2040 g$=INKEY$
2050 IF g$<>"" THEN EXIT get_loop
2060 END REPeat get_loop
2070 IF g$="y" OR g$="Y" THEN assign_
object
2080 CLS#11
2090 END DEFine describe
```

Comentário sobre «Procdescribe»

Esta rotina processa o *input* introduzido pelo utilizador em resposta às indicações do computador, chamando um par de outras rotinas, *Procassign_mon* e *Procassign_object*. A descrição de cada cela («um túnel frio e escuro», «uma sala do trono rodeada por reposteiros de veludo», «uma cratera lunar fria e sem ar», ou o que quer que seja que agrade ao utilizador) é colocada em *map % (linha 1930)*. As variáveis *vp* e *hp* asseguram que o programa não troque as descrições das várias celas. Se o utilizador quiser mudar uma parte de uma descrição de uma cela, tem de repetir todas as informações sobre essa cela.


```
2360 DEFine PROCedure assign_mon
2370 FOR i= 1 TO 16
2380 PRINT #13,i;" ";monster$(i)
2390 NEXT i
2400 CLS#11
2410 PRINT #11;"Type the number of th
e monster"
2420 INPUT #11, map%(vp,hp,3)
2430 CLS #11
2440 PRINT #11;"How many ";monster$(m
ap%(vp,hp,3));"s in this room? (maxim
um 3)"
2450 INPUT #11,map%(vp,hp,4)
2455 IF map%(vp,hp,4)>3 THEN map%(vp,
hp,4)=3
2460 END DEFine assign_mon
```

Comentário sobre «Procassign_mon»

Esta rotina imprime, na janela # 13, uma lista dos monstros disponíveis (há dezasseis — ver as linhas de dados 12790-12800). Na janela # 11, destinada às mensagens e às indicações, é pedido ao utilizador que introduza o número do monstro *(linhas 2370-2380)*. Esta informação é colocada em *map %* (no terceiro elemento deste quadro) e, seguidamente, é pedido ao utilizador que diga quantos monstros de cada tipo são necessários (esta informação vai para o quarto elemento de *map %*). Só pode haver um tipo de monstro em cada cela. Não se esqueça de que *vp* e *hp* são as coordenadas da cela de que se está a tratar.

PROCASSIGN_OBJECT

```
2470 DEFine PROCedure assign_object
2480 CLS#11
2490 CLS#13
2500 PRINT #11;"Which object in this
room?"
```

```
2510 PRINT #11;"(Type name OR number)
"
2520 INPUT#11; temp$
2530 IF LEN(temp$)>2 THEN
2540 name_dec
2550 ELSE
2560 num_dec
2570 END IF
2580 IF t%=0 THEN
2590 PRINT #11;"No such object"
2600 PAUSE 60
2610 ELSE
2620 map%(vp,hp,2)=t%
2630 END IF
2640 END DEFine assign_object
```

Comentário sobre «Procassign_object»

Esta rotina faz mais ou menos o mesmo que *Procassign_Mon,* mas desta vez trata de objectos em vez de monstros. Neste caso não é apresentada a lista, pois, como temos cerca de trinta e seis objectos (linhas de dados 12430-12780), se quiséssemos mostrá-los todos ao mesmo tempo teríamos de utilizar corpo 8. Assim, neste caso, o programa aceita tanto o número quanto o nome do objecto. Só pode existir um objecto por cela.

Linhas 2520-2620: Estas linhas colocam o *input* numa cadeia temporária, *temp$*. Se o comprimento desta cadeia for *mais longo* que 2 *(linha 2530),* parte-se do princípio de que é um nome, sendo chamado *Procname_dec;* caso contrário, é chamado *Procnum_dec.* O resultado da chamada de qualquer destas rotinas (que vão ser discutidas quando chegar a sua vez) é colocado em *map% (linha 2620).*

PROCNUM_DEC

```
2650 DEFine PROCedure num_dec
2660 t%=temp$
2670 IF t%<1 OR t%>36 THEN t%=0
2680 END DEFine num_dec
```


Comentário sobre «Procnum_dec»

Esta rotina atribui a *t %* o número do monstro ou do objecto que constituem o *input* do utilizador.

PROCNAME_DEC

```
2690 DEFine PROCedure name_dec
2700 t%=0
2710 FOR j=1 TO 36
2720 IF temp$ INSTR object$(j,2) THEN
 t%=j
2730 NEXT j
2740 END DEFine name_dec
```

Comentário sobre «Procname_dec»

Esta rotina verifica a existência do nome do objecto em *object$*. Se o *input* for válido, a linha 2620 de *Procassign_object* coloca o seu valor em *map %*. Se *não* for válido, a função INSTR, na *linha 2720* (consultar o manual, *Keywords*), atribui a *t %* o valor 0, sendo este transmitido a *Procassign-object,* linha 2580, que lhe dá o tratamento conveniente.

Agora já está completa a definição inicial do mapa da aventura: temos uma quadrícula de 12×24 celas ou divisões, tendo cada uma delas uma selecção de saídas, monstros e objectos de acordo com os desejos do utilizador. Assim, voltamos às linhas 230-250, no ciclo supervisor (no programa controlador). Estas linhas são activadas por uma pressão na tecla «3» (opção 3, linha 590, em *Procmenu*), que nos leva para a rotina que vai alterar os dados dos objectos. Nesta altura é chamado *Procobject.*

PROCOBJECT

```
4000 DEFine PROCedure object
4010 REPeat choose
4020 CLS
```

```
4030 AT 5,4:PRINT"Do you want to:"
4040 PRINT "        change object data
[1-18]  (1)"
4050 PRINT "             or object data
[19-36] (2)"
4060 PRINT"             or descriptions
      (3)"
4070 PRINT "             or end routine
      (4)"
4080 get "1","4"
4090 g=g$
4100 SELect ON g
4110 ON g=1,2
4120 obdat
4130 ON g=3
4140 obdesc
4150 ON g=4
4160 EXIT choose
4170 END SELect
4180 END REPeat choose
4190 END DEFine object
```

Comentário sobre «Procobject»

Esta rotina começa por imprimir um minimenu. Deve aqui notar-se que é importante obter as linhas de impressão afastadas das margens do visor, como estas, pois junto da margem elas podem ser perdidas com muita facilidade.

Linha 4080: Procget é chamada de novo, para estabelecer limites entre 1 e 4, e, seguidamente, dependendo do número escolhido, é chamada *Procobdat* (1 e 2), *Procobdesc* (3) ou sai-se da rotina (4). A rotina de modificação de dados sobre objectos ocupa o visor por duas vezes sucessivas, pois há muitos objectos de que é necessário tratar.

PROCOBDAT

```
4200 DEFine PROCedure obdat
4210 kon=g
4220 map_screen kon,3,1,6,18,2
4230 vp=2:hp=9
4240 ov=vp:oh=hp
4250 REPeat control
4260 g$=INKEY$
4270 g=CODE(g$)
4280 SELect ON g
4290 ON g= 48 TO 57
4300 object%(vp-1+((kon-1)*18),(hp/3)
-2)=g-48
4310 ON g= 192
4320 hp=hp-3
4330 IF hp<9 THEN hp=9
4340 ON g= 200
4350 hp=hp+3
4360 IF hp>24 THEN hp=24
4370 ON g=208
4380 vp=vp-1
4390 IF vp<2 THEN vp=2
4400 ON g=216
4410 vp=vp+1
4420 IF vp>19 THEN vp=19
4430 ON g=232
4440 EXIT control
4450 END SELect
4460 AT #10,ov,oh:INK #10,0:PRINT #10
,object%(ov-1+((kon-1)*18),(oh DIV 3)
-2)
4470 AT #10,vp,hp:INK #10,7:PRINT #10
;object%(vp-1+((kon-1)*18),(hp DIV 3)
-2):ov=vp:oh=hp
4480 END REPeat control
4490 END DEFine obdat
```

Comentário sobre «Procobdat»

A *linha 4210* estabelece uma variável local, *kon* e, seguidamente, chama *Procmap_screen*. Ainda não tratámos desta rotina, mas o leitor vai ver que ela altera os valores da *spreadsheet* (neste caso a *spreadsheet* do objecto), de acordo com as instruções do utilizador. As *linhas 4280-4430* lêem as pressões nas teclas. Na *spreadsheet* estabelecida e apresentada em *Procmap_screen*, a posição inicial do cursor é o canto superior esquerdo. O cursor toma a forma de um número brilhante, ou seja, o número que se encontra no canto superior esquerdo brilha. Se for lida uma tecla do teclado, o cursor move-se nessa direcção, e o número que se encontra na nova posição do cursor brilha. Se teclar um número de 0 a 9 (só é permitido um dígito — os números de código correspondentes são 48-57, ver *linha 4290*), esse número é substituído na *spreadsheet*, cujos valores originais aí foram introduzidos a partir das declarações de dados que se encontram no fim do programa (finalmente, *Procmap_screen* vai colocar esses valores em *object%*).

PROCOBDESC

```
4500 DEFine PROCedure obdesc
4510 disob
4520 messob
4530 comob
4540 REPeat loop
4550 CLS#12
4560 INK #12,2
4570 PRINT #12,"Type number and press
 <ENTER>":INPUT#12; number
4580 IF number<1 OR number>36 THEN EN
D REPeat loop
4590 x=(number-1) MOD 12:y= ((number-
1) DIV 12)*12
4600 AT #10,x+1,y+3:INK #10,1:PRINT#1
0;object$(number,2)
4610 CLS#11:PRINT#11,object$(number,1
);" ";object$(number,2);" ";object$(n
umber,3)
4620 REPeat comloop
4630 PRINT#12,"B = change text Before
 keyword"
4640 PRINT#12,"A = change text After
keyword"
4650 PRINT #12,"N = Next keyword (sto
res current text)"
4660 PRINT #12,"E = End choice"
4670 get "A","n"
4680 g= CODE(g$)
4690 SELect ON g
4700 PRINT #11,g
4710 ON g=65,97
4720 after
4730 ON g=66,98
4740 before
4750 ON g=69,101
4760 EXIT loop
4770 ON g=110,78
4780 EXIT comloop
4790 END SELect
4800 END REPeat comloop
4810 CLS#11:AT #10,x+1,y+3:INK #10,2:
PRINT#10;object$(number,2)
4820 END REPeat loop
4830 END DEFine obdesc
```

Comentário sobre «Procobdesc»

Não só é possível alterar as várias propriedades dos objectos, como também é possível alterar as suas descrições. É disso que esta rotina trata. Esta rotina vai, por sua vez, chamar outras três rotinas *(Procdisob, Procmessob* e *Proccomob)*. Na janela # 2 é impresso o nome do objecto e, seguidamente, o minimenu *(linhas 4630-4660)*, que dá ao jogador quatro escolhas. Deve notar-se que, para acabar a rotina, é necessário seleccionar um objecto e, seguidamente, pre-

mir E para acabar (premir apenas E não é suficiente). A nossa velha amiga *Procget* é chamada outra vez nesta ocasião, sendo também chamadas mais duas rotinas, *Procbefore* e *Procafter (linhas 4710-4740).*

PROCDISOB

```
4840 DEFine PROCedure disob
4850 CLS
4860 OPEN #10,scr_464x134a16x16
4870 PAPER#10,3
4880 INK #10,0
4890 CLS#10
4900 BORDER #10,2,1
4910 PRINT #10,"   Name            Name
   Name"
5010 FOR i = 1 TO 12
5020 AT#10,i,0:IF i<10 THEN PRINT#10,
"0";:AT #10,i,1:PRINT #10!i!object$(i
,2)!
5030 IF i>9 THEN AT#10,i,0:PRINT#10!i
!object$(i,2)!
5040 AT#10,i,12:PRINT#10!i+12!object$
(i+12,2)!
5050 AT#10,i,24:PRINT#10!i+24!object$
(i+24,2)!
5060 NEXT i
5070 END DEFine disob
```

Comentário sobre «Procdisob»

Esta rotina apresenta uma lista de todos os objectos na janela # 10.

PROCMESSOB

```
5080 DEFine PROCedure messob
5090 OPEN#11,scr_464x32a16x150
5100 PAPER#11,0
5110 INK #11,7
5120 CLS#11
5130 BORDER#11,3,5
5140 END DEFine messob
```

Comentário sobre «Procmessob»

Esta rotina abre uma janela, # 11, para *input* em texto. Deve notar-se a pequena diferença existente entre os primeiros parâmetros de definição de cada janela nesta rotina e na seguinte, *Proccomob*. Na *linha 5090,* o primeiro parâmetro é «scr» (*screen,* visor), enquanto o primeiro parâmetro na linha 5160, em *Proccomob,* é «con» (*console,* consola). A primeira destina-se apenas a apresentar informação ao utilizador, enquanto a segunda é para o seu *input* através do teclado.

PROCCOMOB

```
5150 DEFine PROCedure comob
5160 OPEN #12,con_464x64a16x182
5170 PAPER#12,7
5180 CLS#12
5190 BORDER#12,3,1
5200 END DEFine comob
```

Comentário sobre «Proccomob»

Esta rotina define uma janela, # 12, e permite que aí apareça *input* (note-se a declaração «con» na *linha 5160*).

PROCBEFORE

```
5210 DEFine PROCedure before
5220 in_text
5230 object$(number,1)=text$
5240 CLS#11
5250 PRINT #11,object$(number,1);" ";
object$(number,2);" ";object$(number,
3)
5260 END DEFine before
```

Comentário sobre «Procbefore»

Esta rotina coloca o *input* (a descrição) em *object$* (número 1), de forma que ele apareça *antes* do nome do objecto.

PROCAFTER

```
5270 DEFine PROCedure after
5280 in_text
5290 object$(number,3)=text$
5300 CLS#11
5310 PRINT #11,object$(number,1);" ";
object$(number,2);" ";object$(number,
3)
5320 END DEFine after
```

Comentário sobre «Procafter»

Esta rotina faz o mesmo que *Procbefore,* mas coloca a descrição *após* o nome do objecto.

PROCIN_TEXT

```
5330 DEFine PROCedure in_text
5340 CLS#12
5350 AT #12,2,24:PRINT #12,"_________
___"
```



```
5355 AT #12,2,0:INPUT #12;"Type the d
escription ",text$
5360 END DEFine in_text
```

Comentário sobre «Procin_text»

Cada uma das rotinas precedentes chama esta pequena rotina, que indica ao utilizador que deve introduzir o seu texto.

O programa volta ao menu principal se a tecla F1 (g = 232) for premida na linha 4430, em *Procobdat*. Agora, o utilizador pode continuar, alterando os dados sobre os monstros ou sobre as personagens.

PROCMONSTER

```
6000 DEFine PROCedure monster
6010 map_screen 3,3,1,6,16,4
6020 vp=4:hp=9:ov=vp:oh=hp
6030 ov=vp:oh=hp
6040 REPeat control
6050 g$=INKEY$
6060 g=CODE(g$)
6070 SELect ON g
6080 ON g= 48 TO 57
6090 monster%(vp-3,(hp/3)-2)=g-48
6100 ON g= 192
6110 hp=hp-3
6120 IF hp<9 THEN hp=9
6130 ON g= 200
6140 hp=hp+3
6150 IF hp>24 THEN hp=24
6160 ON g=208
6170 vp=vp-1
6180 IF vp<4 THEN vp=4
6190 ON g=216
```

```
6200 vp=vp+1
6210 IF vp>19 THEN vp=19
6220 ON g=232
6230 EXIT control
6240 END SELect
6250 AT #10,ov,oh:INK #10,0:PRINT #10
,monster%(ov-3,(oh DIV 3)-2)
6260 AT #10,vp,hp:INK #10,7:PRINT #10
;monster%(vp-3,(hp DIV 3)-2):ov=vp:oh
=hp
6270 END REPeat control
6280 END DEFine monster
```

Comentários sobre «Procmonster»

Esta rotina vai alterar os *stats* dos nossos monstros, e é chamada a partir do ciclo supervisor e do menu de abertura (opção 4) pela linha 280, no controlador. Tal como acontece em *Procobdat*, esta rotina passa antes de mais nada os seus valores para *Procmap_screen* (na *linha 6010*). As restantes linhas consistem mais ou menos na mesma coisa que em *Procobdat*, com a diferença de que os valores agora atribuídos pelo utilizador são colocados em *monster% (linhas 6250-6260)*.

PROCCHARACTER

```
7000 DEFine PROCedure character
7010 map_screen 4,3,1,5,6,4
7020 vp=6:hp=9
7030 ov=vp:oh=hp
7040 REPeat control
7050 g$=INKEY$
7060 g=CODE(g$)
7070 SELect ON g
7080 ON g=78,110
7090 name vp
7100 ON g= 48 TO 57
```



```
7110 character%(vp-5,(hp/3)-2)=g-48
7120 ON g= 192
7130 hp=hp-3
7140 IF hp<9 THEN hp=9
7150 ON g= 200
7160 hp=hp+3
7170 IF hp>21 THEN hp=21
7180 ON g=208
7190 vp=vp-1
7200 IF vp<6 THEN vp=6
7210 ON g=216
7220 vp=vp+1
7230 IF vp>11 THEN vp=11
7240 ON g=232
7250 EXIT control
7260 END SELect
7270 AT #10,ov,oh:INK #10,0:PRINT #10
,character%(ov-5,(oh DIV 3)-2)
7280 AT #10,vp,hp:INK #10,7:PRINT #10
;character%(vp-5,(hp DIV 3)-2):ov=vp:
oh=hp
7290 END REPeat control
7300 END DEFine character
```

Comentário sobre «Proccharacter»

Se o utilizador optar por mudar os dados de uma personagem, pode fazê-lo premindo a tecla 5 do menu de abertura, o que leva a que a linha 310, no controlador, chame esta rotina. Mais uma vez se trata de uma rotina muito semelhante a *Procobdat*, que transporta os seus prórprios valores para *Procmap_screen*. Os valores alterados são então colocados em *character%* (nas *linhas 7270-7280*). Os nomes das personagens também podem ser mudados *(linhas 7080-7090* e *Procname)*.

PROCNAME

```
11000 DEFine PROCedure name(pos)
11010 AT#10,pos,0:PRINT #10,"________
"
11020 initial$=""
11030 FOR i=1 TO 6
11040 initial$=character$(i,1)&initia
l$
11050 END FOR i
11060 count=1
11070 name$=""
11080 REPeat loop
11090 REPeat gloop
11100 g$=INKEY$
11110 IF g$<>"" THEN EXIT gloop
11120 END REPeat gloop
11130 IF CODE(g$)>90 THEN g$=CHR$(COD
E(g$)-32)
11140 IF count =1 AND (g$ INSTR initi
al$<>0) THEN AT #10,18,0:PRINT#10, "I
llegal character":EXIT loop
11150 AT #10,pos,count-1:PRINT#10,g$
11160 name$=name$ & g$
11170 IF g$=CHR$(10) OR count>7 THEN
EXIT loop
11180 count=count+1
11190 END REPeat loop
11200 IF count<8 THEN
11210 AT#10,pos,count-1:PRINT #10;FIL
L$(" ",(9-count))
11220 END IF
11230 character$(pos-5)=name$
11240 END DEFine name
```


Comentário sobre «Procname»

Sendo agora apresentadas ao utilizador as *spreadsheats* das personagens, este pode mudar-lhes os nomes que tinham sido inicialmente definidos nas declarações de dados colocadas no fim do programa. Premindo N (de Nome) — ver *Procmap_screen* —, o nome que está a ser apresentado na posição do cursor desaparece, e são criados espaços para que aí seja teclado o novo nome *(linha 11010)*.

PROCMAP_SCREEN

```
10000 DEFine PROCedure map_screen(con
trol,cellh,cellv,toth,totv,totcat)
10010 OPEN #10,scr_432x216a32x8
10020 BORDER #10,1,0
10030 PAPER #10,3
10040 INK#10,0
10050 CLS#10
10060 CSIZE #10,0,0
10070 OPEN #11,SCR_432X32A32X224
10080 CLS #11
10090 PRINT #11;"Cursor keys move cur
sor."!"Change a value by overtyping i
t."!"F1 to end routine.";
10095 IF control=4 THEN PRINT #11," P
ress N to change name"
10100 FOR i= 1 TO totv
10110 SELect ON control
10120 ON control = 1
10130 AT #10,(i*cellv)+1,0;:PRINT #10
;object$(i,2)
10140 ON control = 2
10150 AT #10,(i*cellv)+1,0;:PRINT #10
;object$(i+18,2)
10160 ON control = 3
10170 AT #10,(i*cellv)+3,0;:PRINT #10
,monster$(i)
```

```
10180 ON control = 4
10190 AT #10,i*cellv+5,0;:PRINT #10;c
haracter$(i)
10200 END SELect
10210 NEXT i
10220 FOR i = 1 TO toth
10230 FOR j=1 TO totcat
10240 SELect ON control
10250 ON control=3,4
10260 AT #10,j-1,(i*cellh)+6;:PRINT #
10,category$(i,j)
10270 ON control=1,2
10280 AT #10,j-1,(i*cellh)+6;:PRINT #
10,axis$(i,j)
10290 END SELect
10300 NEXT j
10310 NEXT i
10320 FOR i=1 TO totv
10330 FOR j=1 TO toth
10340 SELect ON control
10350 ON control=1
10360 AT #10,(i*cellv)+1,(j*cellh)+6;
: PRINT #10;object%(i,j)
10370 ON control=2
10380 AT #10,(i*cellv)+1,(j*cellh)+6;
: PRINT #10;object%(i+18,j)
10390 ON control=3
10400 AT #10,(i*cellv)+3,(j*cellh)+6;
: PRINT #10;monster%(i,j)
10410 ON control=4
10420 AT #10,(i*cellv)+5,(j*cellh)+6;
: PRINT #10;character%(i,j)
10430 END SELect
10440 NEXT j
10450 NEXT i
10460 END DEFine map_screen
```


Comentário sobre «Procmap_screen»

Esta é a rotina que é utilizada por cada uma das rotinas seguintes: *Procobject, Procmonster* e *Proccharacter*. O primeiro parâmetro, na *linha 10000,* recebe o seu valor inicial a partir da linha 4220, em *Procobdat,* da linha 6010, em *Procmonster,* e da linha 7010, em *Proccharacter*. Os outros parâmetros determinam o espaçamento horizontal e vertical da quadrícula («cellh» e «cellv»), os incrementos horizontal e vertical («toth» e «totv»). As nove categorias (que se encontram na linha de dados 12810) são apresentadas ao longo do topo da *spreadsheet*. Seguidamente, é definida uma janela, # 10, em que vai ser apresentada a *spreadsheet* e outra janela, # 11, em que vai ser impresso o menu de indicações no utilizador *(linha 10090).*

Dependendo da rotina a partir da qual *Procmap_screen* foi chamada, o programa vai agora imprimir, como é pedido, o conteúdo das várias cadeias sob a forma de uma quadrícula: do lado esquerdo vão ficar impressos os nomes dos monstros, objectos ou personagens, enquanto em cima vão ficar as categorias (referentes a pontos de *Attack, Defence* e *Spell,* etc.).

Já tratámos de todas as escolhas que o menu principal põe à disposição do utilizador, exceptuando duas «facilidades» importantes: as opções 1 e 6. A opção 1 serve para introduzir dados a partir de um *microdrive* onde foram arquivados durante uma sessão anterior com o Gerador.

Porém, de momento, faz mais sentido tratar da opção 6, que serve para acabar o programa. Antes de fechar completamente, o programa permite ao utilizador salvaguardar todos estes dados tão arduamente obtidos.

PROCFILE_OUT

```
9000 DEFine PROCedure file_out
9010 PRINT "Please type name of game"
9020 INPUT game$
9030 PRINT"Place the cartridge into M
icrodrive 1"
```

```
9040 PRINT "And press <Enter>"
9050 REPeat get_loop
9060 in$=INKEY$
9070 IF in$=CHR$(10) THEN EXIT get_lo
op
9080 END REPeat get_loop
9090 gn$="MDV1_"& game$ & "numbers"
9100 OPEN_NEW #9, gn$
9110 FOR i=1 TO 36
9120 FOR j=1 TO 6
9130 PRINT #9,object%(i,j)
9140 NEXT j
9150 NEXT i
9160 FOR i=1 TO 12
9170 FOR j=1 TO 24
9180 FOR k=1 TO 4
9190 PRINT #9,map%(i,j,k)
9200 NEXT k
9210 NEXT j
9220 NEXT i
9230 FOR i=1 TO 16
9240 FOR j=1 TO 6
9250 PRINT #9,monster%(i,j)
9260 NEXT j
9270 NEXT i
9280 FOR i=1 TO 6
9290 FOR j=1 TO 5
9300 PRINT #9,character%(i,j)
9310 NEXT j
9320 NEXT i
9322 PRINT#9,turns
9323 PRINT#9,monsterskilled
9324 PRINT#9,v
9325 PRINT#9,h
9330 CLOSE #9
9340 gm$="mdv1_"&game$&"names"
9350 OPEN_NEW #9, gm$
9360 FOR i= 1 TO 12
9370 FOR j=1 TO 24
```



```
9380 PRINT #9,map$(i,j)
9390 NEXT j
9400 NEXT i
9410 FOR i= 1 TO 36
9420 FOR j=1 TO 3
9430 PRINT #9,object$(i,j)
9440 NEXT j
9450 NEXT i
9460 FOR i =1 TO 6
9470 PRINT #9,character$(i)
9480 NEXT i
9485 PRINT #9,char$
9490 CLOSE #9
9500 END DEFine file_out
```

Comentário sobre «Procfile__out»

Esta rotina leva muito tempo a teclar. Contudo, não faz mais que abrir um canal *(linhas 9100* e *9350)* e, em seguida, através desse canal, imprimir *(PRINT)* todos os dados na *cartridge.* Cada cadeia e variável tem de ser lida por sua vez — o programa fornece automaticamente os nomes de arquivo.

De volta ao menu principal, o utilizador pode escolher a opção 1, que irá introduzir dados obtidos numa sessão anterior com o QLAG.

PROCINFILE

```
8000 DEFine PROCedure infile
8010 CLS
8020 PRINT "Load which game?"
8030 INPUT game$
8040 PRINT"Place the cartridge into M
icrodrive 1"
8050 PRINT "And press <Enter>"
8060 REPeat get_loop
```

```
8070 in$=INKEY$
8080 IF in$=CHR$(10) THEN EXIT get_lo
op
8090 END REPeat get_loop
8100 gn$="mdv1_"&game$&"numbers"
8110 gm$="mdv1_"&game$&"names"
8120 OPEN #9, gn$
8130 FOR i=1 TO 36
8140 FOR j=1 TO 6
8150 INPUT #9,object%(i,j)
8160 NEXT j
8170 NEXT i
8180 FOR i=1 TO 12
8190 FOR j=1 TO 24
8200 FOR k=1 TO 4
8210 INPUT #9,map%(i,j,k)
8220 NEXT k
8230 NEXT j
8240 NEXT i
8250 FOR i=1 TO 16
8260 FOR j=1 TO 6
8270 INPUT #9,monster%(i,j)
8280 NEXT j
8290 NEXT i
8300 FOR i=1 TO 6
8310 FOR j=1 TO 5
8320 INPUT #9,character%(i,j)
8330 NEXT j
8340 NEXT i
8342 INPUT#9,turns
8343 INPUT#9,monsterskilled
8344 INPUT#9,v
8345 INPUT #9,h
8350 CLOSE #9
8360 OPEN #9, gm$
8370 FOR i= 1 TO 12
8380 FOR j=1 TO 24
8390 INPUT #9,map$(i,j)
8400 NEXT j
```



```
8410 NEXT i
8420 FOR i= 1 TO 36
8430 FOR j=1 TO 3
8440 INPUT #9,object$(i,j)
8450 NEXT j
8460 NEXT i
8470 FOR i =1 TO 6
8480 INPUT #9,character$(i)
8490 NEXT i
8495 INPUT#9,char$
8500 CLOSE #9
8510 END DEFine infile
```

Comentário sobre «Procinfile»

Esta rotina é a inversa de *Procfile_out:* nela, a informação desloca-se em sentido contrário. Os dados são salvaguardados e introduzidos em duas partes: «números», ou seja, as variáveis contendo informação numérica, e «nomes», ou seja, as cadeias contendo os vários textos.

DECLARAÇÕES DE DADOS

```
12430 DATA "sword",1,1,1,1,1,1
12440 DATA "torch",2,2,2,2,2,2
12450 DATA "club",3,3,3,3,3,3
12460 DATA "axe",2,2,2,2,2,2
12470 DATA "knife",2,5,5,5,5,5
12480 DATA "staff",3,3,3,3,3,3
12490 DATA "book",4,4,4,4,4,4
12500 DATA "coin",7,7,7,7,7,7
12510 DATA "orb",3,4,3,4,3,4
12520 DATA "flower",5,5,5,5,5,8
12530 DATA "wand",6,6,6,6,6,6
12535 DATA "gem",4,5,6,8,6,5
12540 DATA "ring",5,5,5,7,7,7
12550 DATA "cloak",7,7,7,7,7,7
```

```
12560 DATA "torque",8,8,8,8,8,8
12570 DATA "helmet",7,6,7,8,7,3
12580 DATA "scroll",9,9,9,9,9,9
12590 DATA "rug",7,6,7,6,5,6
12600 DATA "shield",7,6,5,4,3,2
12610 DATA "stone",8,7,6,5,4,3
12620 DATA "bone",4,5,6,7,8,9
12630 DATA "boots",9,8,7,6,5,4
12640 DATA "bow",6,5,4,7,6,5
12650 DATA "box",3,3,3,4,4,4
12660 DATA "sack",5,6,7,8,9,0
12670 DATA "hat",5,4,3,6,5,4
12680 DATA "cross",5,4,6,7,8,9
12690 DATA "goblet",4,5,6,7,8,9
12700 DATA "crown",5,4,3,6,5,4
12710 DATA "sceptre",2,4,6,8,9,7
12730 DATA "belt",4,5,6,8,6,5
12740 DATA "jar",4,3,6,5,1,6
12750 DATA "bracelet",7,4,6,3,5,2
12760 DATA "feather",7,5,3,6,4,2
12770 DATA "bottle",5,4,3,7,5,7
12780 DATA "potion",7,5,3,4,6,8
12790 DATA "snake","ghost","wild dog"
,"roper","gnome","dragon","spider","g
oblin","troll"
12800 DATA "wizard","hulk","zombie","
ostrich","vulture","demon","lionman"
12810 DATA "ATTK","DFNC","SPEL","ANGR
","NERV","GRED","CURS","LOCN","BANE"
12820 DATA "GAEYLOR","ARAGAUNT","GRUM
P","STUGGLE","ENDAR","SHELL"
12830 DATA "AT","DF","SP","CS","LC","
BN"
```

Comentário sobre as declarações de dados

Chegámos finalmente aos dados a ser utilizados no programa. Em comparação com o próximo programa, QAD, as linhas não são muitas. Os nomes que se encontram na *linha 12820* não são importantes, pois podem ser modificados à vontade: o que que quer que o utilizador dactilografe pode ser modificado no decurso do programa. As outras informações de dados só podem ser alteradas quando o utilizador introduz as linhas de dados. Os números que se encontram nas *linhas 12430-12780* aparecem nas *spreadsheets* dos objectos.

CAPÍTULO 7

Um jogo de aventuras no «QL»: «QAD»

Uma vez que a totalidade das linhas para o programa gerador esteja escrita, vamos salvaguardar o programa em *microdrive*. Podemos agora introduzir o programa que vai passar todos os dados gerados por QLAG — vai ser este o nosso programa de aventuras, a que vamos chamar QAD (QL Adventure).

Ao chegar a este ponto, temos de tomar uma decisão: queremos uma aventura com imagens ou uma aventura apenas em texto? Se queremos imagens (a aventura com imagens vai decerto ser muito compensadora), temos de incluir todas as rotinas referentes a imagens (Procmonster_show, Procobj_show, Procdoor_draw e Procgraphic). Além disso, o leitor vai ter uma trabalheira infernal a teclar todas as linhas de dados que se encontram no final do programa, ainda que vá ver que, ao princípio, não precisa de as introduzir na totalidade: basta um par delas para ter um cheirinho do jogo. No próximo capítulo vamos dar-lhe um pequeno programa para definir as suas próprias imagens, de modo que o fardo seja um pouco aligeirado. Se se decidir por uma aventura só com texto, tem de se certificar de que não vai chamar as rotinas referentes a imagens a partir de outras rotinas.

Mas prossigamos com QAD, o programa que vai passar usando os dados estabelecidos por nós no primeiro programa, QLAG.

ESTABELECIMENTO E CICLO PRINCIPAL

```
100 REMark QL ADVENTURE  1984
 N. RICHARD WILLIAMS & TONY BRIDGE
110 CLEAR
120 RESTORE
130 init
140 infile
150 char_select
160 cell_display
170 char_check char$
180 REPeat main_loop
190 CLS#14
200 PRINT #14,"B=battle, C=chat, D=dr
ink, G=give, I=investigate, L=leave,
M=move, Q=quit, R=rest, S=score, T=ta
ke"
210 REPeat getloop
220 keypress$=INKEY$
230 keypress=CODE(keypress$)
240 IF keypress<123 AND keypress>64 T
HEN EXIT getloop
250 END REPeat getloop
260 turns=turns+1
270 qr=RND(1 TO (LEN(char$))):s=RND(1
 TO 2)
280 character%(char$(qr),s)=character
%(char$(qr),s)-1
290 SELect ON keypress
300 ON keypress=77,109
310 char_move
320 ON keypress=76,108
330 drop char$(RND(1 TO 3) )
340 ON keypress=73,105
350 cell_display:char_check char$
360 ON keypress=66,98
370 combat
380 ON keypress=84,116
390 take
400 ON keypress=68,100
410 drink
420 ON keypress=83,115
430 score
440 ON keypress=82,114
450 rest
460 ON keypress=67,99
470 talk
480 ON keypress=71,103
490 give
500 ON keypress=81,113
510 EXIT main_loop
520 END SELect
530 FOR k=1 TO LEN(char$)
535 FOR m=6 TO 8
540 IF object%(character%(char$(k),m)
,5) = 2 THEN victory=1
545 PRINT object%(character%(char$(k)
,m),5)
550 NEXT m
555 NEXT k
560 IF LEN(char$)=0 OR victory=1 THEN
 EXIT main_loop
570 END REPeat main_loop
580 IF victory=1 THEN celebrate:STOP
590 IF LEN(char$)=0 THEN cry:STOP
600 PRINT #14,"Do you want to save da
ta? (Y/N)"
610 REPeat loop
620 g$=INKEY$(0)
630 IF g$="Y" OR g$="N" THEN EXIT loo
p
640 END REPeat loop
650 IF g$="Y" THEN file_out
660 PRINT #14,"Farewell"
670 STOP
```

Comentários sobre «Estabelecimento e ciclo principal»

Esta primeira parte do programa controla a totalidade do que se vai seguir, chamando no princípio várias rotinas. *Procinit* e *Procinfile* já devem ser familiares ao leitor, pois apareceram no capítulo anterior, sobre QLAG.

A estrutura do ciclo principal *(linhas 180-510)* é bastante fácil de seguir. É impresso na janela # 14 um menu contendo várias opções e chamando, de acordo com a escolha do jogador, uma de entre várias rotinas (tenha-se em atenção que o *input* é fornecido tanto em caracteres minúsculos quanto em caracteres maiúsculos, sendo verificado o código de cada um deles).

Battle (batalha)
Chat (conversa)
Drink (beber)
Give (dar)
Investigate (investigar)
Leave (deixar, largar)
Move (andar)
Quit (desistir)
Rest (descansar)
Score (pontuação)
Take (agarrar)

Chama *Proccombat*
Chama *Proctalk*
Chama *Procdrink*
Chama *Procgive*
Chama *Proccell display*
Chama *Procdrop*
Chama *Procchar move*
Sai do *loop* principal
chama *Procrest*
chama *Procscore*
chama *Proctake*

Ao sair do ciclo principal, o programa está de facto acabado, ainda que haja alguns pontos menores a tratar. O computador pergunta ao jogador se quer salvaguardar dados neste jogo: se quiser, é chamado *Procfile_out*, sendo tudo salvaguardado.

Vamos tratar das rotinas na ordem por que são chamadas pelo programa.

PROCINIT

```
6650 DEFine PROCedure init
6660 RESTORE
6670 base=138898
6680 OPEN #5,scr_153x88a32x16
6690 OPEN #6,scr_153x88a185x16
6700 OPEN #7,scr_153x88a337x16
6710 OPEN #8,scr_180x88a32x16
6720 OPEN #9,scr_280x52a212x16
6730 OPEN #10,scr_280x36a212x68
6740 OPEN #11,scr_153x110a32x104
6750 OPEN #12,scr_153x110a185x104
6760 OPEN #13,scr_153x110a337x104
6770 OPEN #14,con_460x42a32x214
6780 FOR i=1 TO 7
6790 INK #(i+7),0
6800 PAPER#(i+7),i-1
6810 CLS #(i+7)
6820 BORDER #(i+7),1,0
6830 NEXT i
6840 INK#8,7
6850 INK#7,0
6860 INK#6,0
6870 INK #5,0
6880 PAPER #5,1
6890 PAPER #6,2
6900 AT #8,3,5:PRINT #8,"ROOM"
```

```
6910 PAPER #7,3
6920 PRINT #10,"CHARACTER MESSAGES"
6930 FOR i=1 TO 3
6940 PRINT #(i+10),"CHARACTER"!" STAT
S"
6950 NEXT i
6960 PRINT #14,"YOUR INPUT"
6970 PRINT #9,"SYSTEM OUTPUT"
6980 h=1:v=1:beforeh=1:beforev=1
6990 dor$="NESW":rdor$="SWNE"
7000 entrance%=4:bargain=0
7010 victory=0
7020 char$="123"
7030 pl=0
7040 comb=0
7050 sc=0
7060 monsterskilled=0:no_of_events=4
7070 turns=0
7080 AT #9,3,2:PRINT #9,"Initialising
...."
7090 DIM map%(12,24,4)
7100 DIM map$(12,24,60)
7110 DIM object$(36,3,12)
7120 DIM object%(36,6)
7130 DIM monster$(16,8)
7140 DIM monster%(16,6)
7150 DIM character%(6,8)
7160 DIM character$(6,8)
7170 DIM category$(9,8)
7180 DIM tempmon%(3,6)
7190 DIM first$(6,20),second$(6,20),n
asty$(6,16)
7200 FOR i=1 TO 6
7210 READ first$(i),second$(i),nasty$
(i)
7220 NEXT i
7230 FOR i=1 TO 16
7240 READ monster$(i)
7250 NEXT i
```



```
7260 FOR i=1 TO 6
7270 READ category$(i)
7280 NEXT i
7290 END DEFine init
```

Comentário sobre «Procinitt»

O endereço de base que se encontra na *linha 6670* é o endereço da parte inferior esquerda da memória do visor, a partir do qual vão ser computadas todas as imagens. Seguidamente, vão ser definidas dez janelas *(linhas 6680-6770),* das quais uma, a janela # 14, vai aceitar *input* através do teclado. Os limites são desenhados e as várias cores da tinta e do papel são estabelecidas. Note-se que cada janela tem uma cor diferente (estabelecida na *linha 6790*). Em cada uma delas é impressa informação sobre o que aí vai ser apresentado no decurso do jogo. Isto dá ao jogador alguma coisa para onde pode ir olhando enquanto se está a fazer a inicialização.

Seguidamente, são atribuídos às variáveis os seus valores iniciais e são dimensionadas várias cadeias. Note-se, nas *linhas 6990--7000,* que as duas cadeias de códigos de entrada («NESW») e códigos de saída («SWNE»), da nossa secção sobre saídas, no cap. 5, são colocadas respectivamente em *dor$* e *rdor$,* e que é atribuído a *entrance%* o valor 4 (que não é um dos valores 0-3). Finalmente, os dados (linhas 9500-11760) são lidos para várias cadeias.

Uma vez estabelecido o que o visor vai apresentar, o programa chama *Procinfile.*

PROCINFILE

```
7300 DEFine PROCedure infile
7310 CLS#14
7320 PRINT #14,"Load which game?"
7330 INPUT #14,game$
7340 PRINT#14,"Place the cartridge in
to Microdrive 1"
7350 PRINT#14, "And press <Enter>"
7360 REPeat get_loop
```

```
7370 in$=INKEY$
7380 IF in$=CHR$(10) THEN EXIT get_lo
op
7390 END REPeat get_loop
7400 gn$="mdv1_"&game$&"numbers"
7410 gm$="mdv1_"&game$&"names"
7420 OPEN #15, gn$
7430 FOR i=1 TO 36
7440 FOR j=1 TO 6
7450 INPUT #15,object%(i,j)
7460 NEXT j
7470 NEXT i
7480 FOR i=1 TO 12
7490 FOR j=1 TO 24
7500 FOR k=1 TO 4
7510 INPUT #15,map%(i,j,k)
7520 NEXT k
7530 NEXT j
7540 NEXT i
7550 FOR i=1 TO 16
7560 FOR j=1 TO 6
7570 INPUT #15,monster%(i,j)
7580 NEXT j
7590 NEXT i
7600 FOR i=1 TO 6
7610 FOR j=1 TO 5
7620 INPUT #15,character%(i,j)
7630 NEXT j
7640 NEXT i
7650 INPUT#15,turns
7660 INPUT#15,monsterskilled
7670 INPUT#15,v
7680 INPUT#15,h
7690 CLOSE #15
7700 OPEN #15, gm$
7710 FOR i= 1 TO 12
7720 FOR j=1 TO 24
7730 INPUT #15,map$(i,j)
7740 NEXT j
7750 NEXT i
7760 FOR i= 1 TO 36
7770 FOR j=1 TO 3
7780 INPUT #15,object$(i,j)
7790 NEXT j
7800 NEXT i
7810 FOR i =1 TO 6
7820 INPUT #15,character$(i)
7830 NEXT i
7840 INPUT#15,char$
7850 CLOSE #15
7860 END DEFine infile
```


Comentário sobre «Procinfile»

Esta rotina vai ser sempre chamada, pois o jogo não pode ser jogado sem um conjunto de dados previamente definidos, transportados de QLAG. As *linhas 7320-7350* dão ao jogador instruções para introduzir os dados para o seu jogo. A rotina segue o mesmo padrão da rotina *Procinfile* do programa QLAG (linhas 8000-8510 de QLAG) — de facto, usamos a mesma rotina, mudando apenas um ou dois pormenores, como o número da janela (que aqui é #15 e não #9, como na rotina pertencente a QLAG).

A melhor maneira de efectuar estas mudanças é salvaguardar (SAVE) o que já se tem de QAD, introduzir seguidamente a instrução NEW e depois introduzir (LOAD) QLAG de novo. Nessa altura, salvaguarde apenas as linhas 8000-8510 de QLAG. Volte a introduzir QAD, misture (MERGE) a rotina, e mude então o que for necessário.

PROCHAR_SELECT

```
680 DEFine PROCedure char_select
690 char$=""
700 CLS#5
710 CLS#6
720 CLS#7
730 BORDER#5,1,0
```

```
740 BORDER #6,1,0
750 BORDER #7,1,0
760 FOR i=1 TO 3
770 FOR j= 1 TO 6
780 PRINT #(i+4),category$(j);" ";cha
racter%(i,j)
790 PRINT #(i+10),category$(j);" ";ch
aracter%(i+3,j)
800 NEXT j
810 AT #(i+4),7,5:PRINT #(i+4),i
820 AT #(i+10),9,5:PRINT #(i+10),i+3
830 NEXT i
840 PRINT #14,"Choose three character
s by pressing      three of the number
s 1 to 6"
850 REPeat count
860 REPeat check
870 g$=INKEY$
880 IF g$ INSTR char$=0 AND g$ INSTR
"123456"<> 0 THEN EXIT check
890 END REPeat check
900 char$=char$ & g$
910 IF LEN(char$)=3 THEN EXIT count
920 END REPeat count
930 END DEFine char_select
```

Comentário sobre «Prochar_select»

Quando os dados seleccionados estão introduzidos, as personagens são apresentadas em seis janelas (números 5, 6, 7, 11, 12 e 13 — *linhas 780-790*), juntamente com os vários parâmetros estabelecidos pelo utilizador em QLAG. O jogador é instruído na janela # 14 para escolher três personagens a fim de entrarem na aventura *(linha 840)*.

As *linhas 850-920* verificam a validade do *input* e retiram a rotina quando tiver sido feita uma escolha válida (para mais informação sobre o operador INSTR, consultar o manual na secção *keywords*).

PROCCELL_DISPLAY

```
940 DEFine PROCedure cell_display
950 CLS#9:BORDER #9,1,0
960 CLS#10:BORDER #10,1,0
970 tell v,h
980 graphic v,h
990 IF map%(v,h,4)<>0 THEN monster_sh
ow
1000 IF map%(v,h,2)<>0 THEN obj_show
1010 END DEFine cell_display
```

Comentário sobre «Proccell_display»

Elimina os conteúdos das janelas # 9 e # 10, define as fronteiras e chama *Proctell* e *Procgraphic*. É feita uma verificação para ver se se encontra nesta cela um objecto ou um monstro (a partir de informação colocada em *map%*). Se estiver presente alguma destas coisas, é chamado o procedimento respectivo *(Procmonster_show* ou *Procobj_show)*. *Procsell_display* também é chamado pelas linhas 340-350 do ciclo principal.

PROCGRAPHIC

```
1020 DEFine PROCedure graphic(v,h)
1030 CLS#8:BORDER#8,1,0
1040 SCALE #8,180,0,0
1050 REMark h gives horizontal and v
vertical map reference
1060 LINE #8,0,0 TO 60,60 TO 60,140 T
O 0,180,270,0 TO 200,60 TO 200,140 TO
 270,180,60,140 TO 200,140,60,60 TO 2
00,60
1070 REMark entrance%=exit entered by
 (n=0,W=3)
1080 REMark if there is a code for th
e directon then draw the door
```

```
1090 PRINT #10,"Exits :";
1100 FOR k=0 TO 3
1110 IF (map%(v,h,1) && 2^k) THEN PRI
NT #10," ";rdor$(k+1);:IF k<>entrance
% THEN door_draw k
1120 NEXT k
1130 PRINT #9," "
1140 END DEFine graphic
```

Comentário sobre «Procgraphic»

Esta é uma rotina referente a imagens. Usando os dois parâmetros *v* e *h,* esta rotina desenha na janela # 8 uma pequena imagem perspectivada da divisão. A *linha 1110* verifica *map%* para ver se há uma porta em alguma das três paredes, usando a função *bitwise* AND e um ciclo (k = 0 TO 3), enquanto o valor de *k* não coincidir com o valor de *entrance%*. Nessa altura é chamado *Procdoor_draw* para desenhar a porta, usando o parâmetro *k*.

PROCMONSTER_SHOW

```
1310 DEFine PROCedure monster_show
1320 PAPER #8,0
1330 FOR mon=0 TO (map%(v,h,4)-1) MOD
 3
1340 REMark number of monsters
1350 RESTORE (9700+(map%(v,h,3)*300))
1360 READ vert_it
1370 READ hor_it
1380 startadd=(mon)*(4+hor_it)
1390 FOR upward=0 TO vert_it
1400 FOR along=0 TO hor_it   STEP 2
1410 READ dat
1420 POKE_W base+2560-(vert_it*128)+s
tartadd+along+(upward*128),dat
1430 END FOR along
1440 END FOR upward
1450 END FOR mon
1460 END DEFine monster_show
```


Comentário sobre «Procmonster_show»

Eis outra rotina referente a imagens. Os dados para a imagem de cada monstro presente no local são lidos, sendo o monstro apresentado no primeiro plano da divisão.

PROCOBJ_SHOW

```
1470 DEFine PROCedure obj_show
1480 REMark draw here obj graphic for
 map%(h,v,2)
1490 RESTORE (14660+(map%(v,h,2)*10))
1500 READ down,along
1510 FOR j=0 TO down
1520 FOR k=0 TO (along) STEP 2
1530 READ a
1540 POKE_W (base+4100+k+(j*128)),a
1550 NEXT k
1560 NEXT j
1570 END DEFine obj_show
```

Comentário sobre «Procobj_show»

Eis outra rotina referente a imagens. Os dados para a imagem de cada objecto presente são lidos, aparecendo o objecto na mesma janela.

PROCTELL

```
1580 DEFine PROCedure tell
1590 CLS#9
1600 CLS#10
1610 PRINT #9,map$(v,h)
1620 IF map%(v,h,4) <>0 THEN PRINT#9,
 "Ahead you see ";map%(v,h,4);" ";:pl
ural
```

```
1630 IF map%(v,h,2)<>0 THEN PRINT#9,
"There is a ";object$(map%(v,h,2));"
nearby."
1640 END DEFine tell
```

Comentário sobre «Proctell»

Esta rotina limita-se a ler *map%*, *map$* e *object$* e a imprimir a informação obtida para o jogador.

PROCDOOR_DRAW

```
1150 DEFine PROCedure door_draw (k)
1160 REMark move the position code
around according to the entrance dire
ction
1170 door_code=(k+4-entrance%) MOD 4
1180 SELect ON door_code
1190 REMark door_code 1= left wall, 2
 = mid wall, 3 = right wall
1200 ON door_code=1
1210 REMark draw left_door
1220 LINE#8,20,20 TO 20,130 TO 40,116
 TO 40,40
1230 ON door_code=2
1240 REMark draw mid_door
1250 LINE #8,110,60 TO 110,115 TO 150
,115 TO 150,60
1260 ON door_code=3
1270 REMark draw right_door
1280 LINE#8,250,20 TO 250,130 TO 230,
116 TO 230,35
1290 END SELect
1300 END DEFine door_draw
```


Comentário sobre «Procdoor_draw»

Ainda mais uma rotina referente a imagens. O parâmetro *k* é o resultado do ciclo *k*, em *Procgraphic* (linhas 1100-1120). Usando-o, a *linha 1170* atribui um código à variável *door_code*, sendo este código usado para decidir em que parede deve ser desenhada a imagem da porta, se aparecer alguma (ver a discussão sobre saídas no cap. 5).

Agora voltamos ao início do programa, onde é chamado *Procchar_check* (linha 350).

PROCCHAR_CHECK

```
1650 DEFine PROCedure char_check(r$)
1660 FOR i=1 TO LEN(r$)
1670 REMark print (window of current
character)
1680 CLS#(i+10)
1690 FOR j=1 TO 5
1700 IF character%(r$(i),j)<1 THEN ch
aracter%(r$(i),j)=0
1710 PRINT #(i+10), category$(j);" ";
character%(r$(i),j)
1720 NEXT j
1730 PRINT #(i+10);"Carrying:"
1740 j=6
1750 jt=0
1760 REPeat loop
1770 IF character%(r$(i),j)<>0 THEN
1780 PRINT#(i+10)," ";object$(charact
er%(r$(i),j),2)
1790 FOR k=3 TO 5
1800 IF object%(character%(r$(i),j),5
)<>1 AND object%(character%(r$(i),j),
5)<>2 THEN character%(r$(i),k)=charac
```

```
ter%(r$(i),k)+object%(character%(r$(i
),j),k):object%(character%(r$(i),j),k
)=0
1810 NEXT k
1820 ELSE
1830 jt=jt+1
1840 END IF
1850 j=j+1
1860 IF j=8 THEN EXIT loop
1870 END REPeat loop
1880 IF jt=2 THEN PRINT#(i+10);" Noth
ing"
1890 AT #(i+10),9,(11-(LEN(character$
(r$(i))))DIV 2:PRINT #(i+10),charact
er$(r$(i))
1900 NEXT i
1910 IF turns>3 AND comb <>1 THEN my_
word
1920 END DEFine char_check
```

Comentário sobre «Procchar_check»

Esta rotina imprime, em três janelas situadas abaixo da área de jogo, os *stats* de cada uma das três personagens previamente escolhidas em *Procchar select,* bem como quaisquer objectos que estas transportem *(linhas 1660-1810)* — claro que, a princípio, não irão transportar nenhuns objectos. Finalmente, após um certo número de vezes (aqui são três, mas isto pode ser alterado se o utilizador quiser) é chamada *Procmy_word* (ver mais adiante).

Agora, o estabelecimento inicial encontra-se efectuado, e é apresentada ao jogador a principal área de exposição mostrando o seu grupo de três personagens. A janela mostra a divisão de que se está a tratar, com as saídas correspondentes e os monstros e objectos que contém. Ao lado encontra-se a janela de informação e sob ela a janela de *input* em texto. Voltamos agora ao ciclo principal do programa, a partir do qual vai ser efectuada a maior parte do trabalho.

PROCCHAR_MOVE

```
1930 DEFine PROCedure char_move
1940 ex%=(entrance%+2)MOD 4
1950 CLS#14
1960 PRINT #14,"Which way?"
1970 REPeat which
1980 REPeat way
1990 g$=INKEY$ (#14)
2000 g$=change$(g$)
2010 entrance%= (g$ INSTR dor$)-1
2020 IF entrance% <>-1 AND g$<>"" THE
N EXIT way
2030 END REPeat way
2040 it= (2^((((g$ INSTR "SsWwNnEe")+
1)DIV 2)-1))
2050 IF (map%(v,h,1)&&it) THEN
2060 EXIT which
2070 ELSE
2080 PRINT #14,"No door that way."
2090 END IF
2100 END REPeat which
2110 IF map%(v,h,4)<>0 AND ex%<>entra
nce% THEN PRINT #9,"You can only go "
;dor$(ex%+1):entrance%=(ex%+2)MOD 4:R
ETurn
2120 beforeh=h:beforev=v
2130 g=CODE(g$)
2140 SELect ON g
2150 ON g=78,110
2160 v=v+1
2170 ON g=83,115
2180 v=v-1
2190 ON g=69,101
2200 h=h+1
2210 ON g=87,119
2220 h=h-1
2230 END SELect
2240 IF h<1 THEN h=1
```

```
2250 IF h>12 THEN h=12
2260 IF v<1 THEN v=1
2270 IF v>24 THEN v=24
2280 cell_display
2290 END DEFine char_move
```

Comentário sobre «Procchar_move»

Esta rotina é chamada pelas linhas 300-310 do ciclo principal. Isto acontece se for seleccionada a opção *move*.

Linha 1940: é dado a *ex%* o valor de código para a porta oposta à entrada actual: se o jogador entrar pela porta Leste é atribuído a *exp%* o valor de Oeste. O jogador introduz então a saída desejada, que é colocada em *entrance% (linha 2010)*. Se, no entanto, existir um monstro na divisão, o jogador só pode sair pela porta por onde entrou, ou seja, só pode retroceder, e é dado de novo a *entrance%* o valor que tinha à entrada (ou seja, o seu valor na *linha 1940*).

Linhas 1980-2030: o ciclo *«way»* repete-se até que seja introduzido um *imput* válido: N, S, E ou W em caracteres minúsculos ou maiúsculos. *Linhas 1970-2100: «way»* é uma parte de um ciclo maior, *«which»* e o programa continua:

Linhas 2110-2230, que pegam no código do carácter introduzido (N, S, E ou W) e actualizam as variáveis das coordenadas *v* (vertical) e *h* (horizontal) conforme for necessário.

Linhas 2240-2270: asseguram que *v* e *h* sejam mantidas entre os limites de 1 e 20. Seguidamente, *Proccell display* é chamada de novo *(linha 2280)*, para uma actualização da apresentação de imagens.

CHANGE$

```
6480 DEFine FuNction change$(g$)
6490 LOCal f$
6500 IF CODE(g$)>90 THEN
6510 f$=CHR$(CODE(g$)-32)
6520 ELSE
6530 f$=g$
```



```
6540 END IF
6550 RETurn f$
6560 END DEFine change$
```

Comentário sobre «Change$»

Esta função é chamada várias vezes em QAD. Isto ocorre pela primeira vez em *Procchar_move*. Utilizamos uma função para repor um valor na rotina a partir da qual a função é chamada. Outras rotinas que usam *change$* são *Procdrink* (linha 2410) e *Procwho* (linha 3490).

A rotina lê o teclado e, mudando o código das teclas, *muda* a resposta em caracteres minúsculos para uma resposta em caracteres maiúsculos. Os códigos dos caracteres minúsculos sã 97-122 e os dos caracteres maiúsculos são 65-90. Para mais pormenores sobre os códigos de tecla, consultar o manual na secção *Concepts*.

PROCDROP

```
8780 DEFine PROCedure drop(dc)
8785 LOCal dp
8790 dp=9
8800 REPeat obloop
8810 dp=dp-1
8820 IF dp=6 OR character%(dc,dp)<>0
 THEN EXIT obloop
8830 END REPeat obloop
8840 IF character%(dc,dp)=0 THEN RETu
rn
8850 IF map%(v,h,2)<>0 THEN PRINT#9,"
The ";object$(map%(v,h,2));" disappea
rs"
8860 map%(v,h,2)=character%(dc,k):PRI
NT #9,map%(v,h,2)
8870 character%(dc,dp)=0
8880 char_check char$:cell_display
8900 END DEFine drop
```

Comentário sobre «Procdrop»

Esta rotina, chamada pelas linhas 320-330 do ciclo principal, permite ao jogador largar um objecto, se lhe apetecer fazê-lo. Na linha 330 do ciclo principal é gerado um número aleatório que é passado para o procedimento que remove *object$* de *map$* e *character%*.

PROCCOMBAT

```
3550 DEFine PROCedure combat
3560 CLS#14
3570 IF map%(v,h,4)<1 THEN PRINT #9,"
No opponents":RETurn
3575 FOR i=1 TO 3:tempmon%(i,1)=0:tem
pmon%(i,2)=0
3580 FOR i=1 TO map%(v,h,4)
3590 FOR j=1 TO 2
3600 tempmon%(i,j)=RND(monster%(map%(
v,h,3),j))+(monster%(map%(v,h,3),j)^2
)
3610 NEXT j
3612 FOR j=3 TO 6
3614 tempmon%(i,j)=monster%(map%(v,h,
3),j)
3616 NEXT j
3620 NEXT i
3630 END FOR i
3640 tchar$=char$ :REMark Create temp
orary character string
3650 REMark Check for bane
3660 banei=0:banej=0
3670 FOR i=1 TO LEN(char$)
3680 FOR j=6 TO 8
3690 IF object%(character%(char$(i),j
),6)=map%(v,h,3) THEN banei=i:banej=j
3700 NEXT j
3710 NEXT i
```



```
3720 IF banei<>0 THEN PRINT #9,"The "
;object$(object%(char%(char$(banei),b
anej),6));" scares them away":map%(v,
h,3)=0:map%(v,h,4)=0:cell_display:RET
urn
3730 REMark If monsters are greedy an
d bargain flag is not set then monste
rs accept gift
3740 greed=0
3750 FOR i=1 TO map%(v,h,4):greed= te
mpmon%(i,5)+greed:NEXT i
3760 IF (greed DIV map%(v,h,4))> RND(
1 TO 9) AND bargain=0 THEN CLS#9:PRIN
T#9,"For a gift you can go past":barg
ain=1:PAUSE 100:RETurn
3770 bargain=0 :REMark Reset bargain
flag
3780 REMark If monsters are very stro
ng or outnumber the party then test e
ach character.
3790 FOR i=1 TO LEN(tchar$)
3800 IF (map%(v,h,4) > LEN(tchar$)) O
R (tempmon%(1,1)>(character%(tchar$(i
),1)) AND RND(1 TO character%(tchar$(
i),5)))>3 THEN
3810 PRINT #9,character$(tchar$(i));"
 runs and hides":CLS #(10+i)
3811 z$=""
3812 FOR k=1 TO LEN(tchar$)
3814 IF tchar$(k)<>ch THEN z$=z$ & tc
har$(k)
3816 END FOR k
3818 tchar$=z$
3820 END IF
3830 NEXT i
3840 REMark If all characters run the
n put them in the previous room and r
eturn
3850 IF LEN(tchar$)=0 THEN v=beforev:
```

```
h=beforeh:RETurn
3860 REMark If either characters or m
onsters are angry then call attack ot
herwise just chat.
3870 angst=0
3880 FOR i= 1 TO LEN(tchar$)
3890 angst = angst + character%(tchar
$(i),4)
3900 NEXT i
3910 angst=angst DIV LEN(tchar$) + RN
D(2)
3920 IF angst>3 THEN
3930 attack
3940 ELSE
3950 PRINT #9,character$(tchar$(1));"
 says: Do we really want to fight?":P
AUSE 100
3960 FOR i=1 TO map%(v,h,4):angmn=ang
mn+tempmon%(i,4):NEXT i
3970 IF angmn>3 THEN
3980 attack
3990 ELSE
4000 PRINT#9,"The ";plural;"replies:
Seems silly to me":PAUSE 100
4010 END IF
4020 END IF
4030 RETurn
4040 END DEFine combat
```

Comentário sobre «Proccombat»

Esta rotina é chamada pelas linhas 360-370 do ciclo principal. Foi esta a primeira parte do programa a ser planeada e, de facto, todo o resto foi gerado a partir deste núcleo. É uma rotina bastante longa, que chama muitas outras rotinas.

Linhas 3560-3570: Eliminam o conteúdo da janela # 14 e verificam em *map%* se de facto existe um monstro no local de que se está a tratar: o jogador devia ficar com cara de parvo a lutar contra

ar e vento. Se não houver oponente, voltamos (RETURN) ao menu principal.

Linhas 3575-3630: Estas linhas eliminam o conteúdo de três janelas situadas no topo do visor. As três janelas inferiores, que apresentam os nomes e *stats* das nossas personagens, mantêm-se durante esta rotina. As três novas janelas vão conter informação sobre os monstros: podem existir até três feras. Seguidamente são atribuídos valores aleatórios a cada uma das cinco categorias na lista de *stats (Attack, Defense,* etc.), sendo impressa essa informação.

Linhas 3640-3720: Agora o programa verifica se está presente um objecto que seja o flagelo *(Bane)* de um monstro. Se ele assustar o monstro (ou os monstros), o programa volta ao menu principal.

Linhas 3730-3770: O factor *Greed* (avidez) do monstro toma inicialmente o valor 0, mas a *linha 3760* compara a variável *greed* com um número aleatório e deixa o jogador passar no caso de ter tido sorte. Neste caso é atribuído o valor 1 a *bargain,* mantendo este valor até a rotina ser chamada de novo. Se o jogador não tiver tido tanta sorte como isso, é atribuído a *bargain* o valor 0 e a rotina continua.

Linhas 3790-3810: Estas linhas comparam as forças relativas dos monstros e do grupo de personagens do jogador. Se as forças dos monstros forem superiores às do grupo de exploração, os membros deste ficam desmoralizados e fogem, voltando à divisão anterior. Contudo, se os monstros ou o grupo estiverem zangados, é chamado *Procattack (linha 3930)* no caso de o valor de *angst* ser baixo; senão, o monstro acaba com a brincadeira e volta-se ao menu principal.

PROCATTACK

```
4090 DEFine PROCedure attack
4100 comb=1:dead=0:retr=0
4110 REMark Whilst a monster exists a
nd a character is in the room
4120 FOR i=1 TO 3:CLS#(i+4):BORDER#(i
+4),1,0:NEXT i
4130 FOR i=1 TO map%(v,h,4):monup(i):
NEXT i
```

```
4140 REPeat while
4150 REPeat choose
4160 mon=RND(1 TO 3)
4170 IF tempmon%(mon,2)<>0 AND tempmo
n%(mon,1) <>0 THEN EXIT choose
4180 END REPeat choose
4190 REPeat chloop
4200 PRINT #14,"Who will attack ";mon
ster$(map%(v,h,3));" ";mon;"?"
4210 who
4220 IF ch INSTR tchar$ <>0 THEN EXIT
 chloop
4230 PRINT#14, "Not here":PAUSE 50
4240 END REPeat chloop
4250 charatt= character%(ch,1)
4260 chardef=character%(ch,2)
4270 charang=character%(ch,4)
4280 FOR j=6 TO 8
4290 IF character%(ch,j)<>0 THEN char
att = charatt + object%(character%(ch
,j),1):chardef=chardef+object%(charac
ter%(ch,j),2)
4300 NEXT j
4310 monatt= tempmon%(mon,1)
4320 mondef=tempmon%(mon,2)
4330 PRINT#14, "Will you :cast a spel
l (S), attack furiously (F), attack c
autiously (C)or retreat (R)?"
4340 REMark Get tactic
4350 tac$=""
4360 REPeat loop
4370 tac$=INKEY$:IF tac$="" THEN GO T
O 4370
4380 IF tac$ INSTR "FfSsCcRr" <>0 THE
N EXIT loop
4390 END REPeat loop
4400 tac=CODE(tac$):hit=0
4410 SELect ON tac
4412 =82,114
```



```
4414 retr=1
4416 EXIT while
4420 =83,115
4430 IF character%(ch,3)>0
4440 PRINT #14,character$(ch);" ";
4450 spell character%(ch,3)
4460 character%(ch,3)=character%(ch,3
)-1
4465 char_check tchar$
4470 ELSE
4480 PRINT#14,"No spells left":PAUSE
100
4490 END IF
4500 =70,102 :frenzy
4510 =67,99 :caution
4520 END SELect
4530 REMark Check for cursed object
4540 chekcurs
4550 IF tac<>83 AND tac<>115 THEN str
ike charatt,mondef
4560 IF kflag=0 AND retr<>1 THEN mons
terhurt(hit)
4565 ELSE PRINT #14,character$(ch);"
is cursed. He hurts himself":PAUSE 10
0
4566 character_hurt hit-1
4567 END IF
4570 REMark Monster chooses tactic
4580 tic=RND(1 TO 3)
4590 hit=0
4600 SELect ON tic
4610 =1
4620 IF tempmon%(mon,3)>0 THEN
4630 PRINT #14,"The ";monster$(map%(v
,h,3));" ";
4640 spell(tempmon%(mon,3))
4650 tempmon%(mon,3)=tempmon%(mon,3)-
1
4660 ELSE
```

```
4670 PRINT #14,"The monster wants to
cast a spell but has run out":PAUSE 1
00
4680 END IF
4690 =2
4700 monatt=monatt+(tempmon%(mon,4) D
IV 2)
4710 mondef=mondef-(tempmon%(mon,4) D
IV 2)
4720 PRINT #14,"The ";monster$(map%(v
,h,3));" is furious":PAUSE 50
4730 =3
4740 monatt=monatt-(tempmon%(mon,6) D
IV 2)
4750 mondef=mondef+(tempmon%(mon,6) D
IV 2)
4760 PRINT #14,"The ";monster$(map%(v
,h,3));" is cautious":PAUSE 50
4770 END SELect
4780 REMark If NOT SPELL then  CALL S
TRIKE for monster
4790 IF tic<>1 THEN strike monatt,cha
rdef
4800 IF hit <>0 THEN characterhurt(hi
t)
4810 REMark If monster is dead then r
emove him from inventory and array
4820 IF tempmon%(mon,2)<1 OR tempmon%
(mon,1)<1 THEN map%(v,h,4)=map%(v,h,4
)-1:CLS#(mon+4):monkil=monkil+1
4830 REMark If character is dead then
 remove him from both temporary and p
ermanent strings
4840 IF character%(ch,1)=0 AND charac
ter%(ch,2)=0 THEN
4850 z$="":x$=""
4860 FOR k= 1 TO LEN(tchar$)
4870 IF tchar$(k)<>ch THEN z$=z$ & tc
har$(k)
```



```
4882 NEXT k
4884 tchar$=z$:char_check tchar$:dead
=1:CLS#(11+LEN(char$))
4890 FOR k=1 TO LEN(char$)
4900 IF char$(k)<>ch THEN x$=x$ & cha
r$(k)
4912 NEXT k
4914 char$=x$
4920 END IF
4930 IF dead=1 OR map%(v,h,4)<1 THEN
EXIT while
4940 END REPeat while
4950 IF map%(v,h,4)=0 THEN
4955 PRINT#14, "All monsters dead":PA
USE 100
4960 ELSE
4962 IF dead=1 THEN
4964 PRINT #14,"A character has died,
 so you run away"
4966 h=beforeh:v=beforev
4967 CLS#(11 + LEN(char$))
4968 ELSE
4970 PRINT #14,"You try to retreat"
4971 IF h<>1 OR v<>1 THEN
4972 h=beforeh:v=beforev
4973 ELSE PRINT #14,"You are trapped"
4974 END IF
4976 END IF
4977 END IF
4978 comb=0:PAUSE 100
4979 cell_display
4980 END DEFine attack
```

Comentário sobre «Procattack»

Esta rotina, que é bastante longa, forma, juntamente com *Proccombat,* o núcleo do presente programa. No entanto, grande parte dela é fácil e não necessita de muitas explicações. Esta rotina

representa sem dúvida um melhoramento em relação à habitual pancadaria, pois apresenta alguma inteligência (a receada «inteligência artificial»). Isto significa que, ainda que o resultado dos combates seja aleatório esta aleatoriedade pode ser influenciada por vários factores. Após a habitual atribuição de valores às variáveis *Comb, Dead,* e *Retr (linha 4100)* é mostrado o monstro (ou os monstros) que se encontram na cela nesse momento. Como o número de monstros pode chegar a três, são estabelecidas três janelas *(linha 4120)* onde são apresentados os *stats* dos monstros. Para isso é chamado *Procmonup*. Depois de perguntar qual das personagens é que vai lutar com o monstro (ou os monstros) o programa chama *Procwho*.

Linhas 4220-4230: Estas linhas evitam qualquer tentativa de nomeação de um membro do grupo que não esteja presente.

Linhas 4250-4290: Estas linhas fixam os valores de *Anger, Defence* e *Attack* da personagem de acordo com o conteúdo de *character%*. Depois de ter impresso o menu de batalha, o programa espera pelo *input* táctico do jogador. Como se pode ver, são apresentadas várias opções ao jogador. Se ele dicidir retroceder, sai-se do *ciclo while,* o que leva o programa a saltar até à *linha 4950;* mas as consequências desta acção nem sempre são vantajosas para o jogador.

No entanto, se o jogador decidir ficar e lutar, é possível usar um feitiço (premindo S de «*spell*») ou atacar, quer com cautela, quer furiosamente.

Linhas 4430-4465: Estas linhas chamam *Procspell* e actualizam o conteúdo da janela de uma personagem após ela ter usado um feitiço, enquanto a *linha 4480* impede as tentativas de usar um feitiço quando a personagem não está de posse de nenhum, levando-a a fazer nova escolha.

Linhas 4500-4510: Estas linhas chamam *Procfrenzy* e *Proccaution,* cujos nomes falam por si (*frenzy* = frenesi, e *caution* = = cautela).

A *linha 4540* chama *Procchekcurs,* que verifica se a personagem está de posse de algum objecto amaldiçoado, que poderia causar-lhe muitos embaraços.

Linhas 4550-4560: Estas linhas chamam *Procstrike* e *Procmonsterhurt,* de que vamos falar daqui a pouco. Seguidamente, é a vez de ser o monstro a distribuir os golpes: é escolhido um número ao acaso *(linha 4580),* que é atribuído à variável *tic* (abreviatura de

tactic, táctica), sendo então escolhida pelo programa, de acordo com este número, uma entre várias opções.

Linhas 4620-4770: Tal como, antes dele, a personagem, o monstro pode fazer uma de três coisas: lançar um feitiço (neste caso é chamado de novo *Procspell*), decidir atacar furiosamente, ou dicidir atacar cautelosamente.

Linhas 4790-4968: Agora o programa vê quem é que morreu em combate. Se ainda ninguém (ou nada) tiver morrido, o programa prossegue com *Procstrike,* actualizando as janelas tanto das personagens quanto dos monstros. Se todos os monstros que se encontravam na divisão já estiverem mortos, voltamos ao menu principal; porém, se tiver morrido uma personagem, o grupo tem de retroceder (*linha 4964*), e os parâmetros de posição são repostos na posição anterior. O grupo pode, evidentemente, chegar à conclusão de que está encurralado *(linha 4973),* sendo a divisão apresentada de novo.

PROCMONUP

```
9080 DEFine PROCedure monup(mon)
9090 CLS#(4+mon)
9100 FOR j=1 TO 5
9110 PRINT #(mon+4),category$(j);" ";
9120 PRINT #(4+mon),tempmon%(mon,j)
9130 END FOR j
9140 AT #(mon+4),7,0:PRINT #(mon+4),m
onster$(map%(v,h,3));" ";mon
9150 END DEFine monup
```

Comentários sobre «Procmonup»

Como já vimos, esta rotina elimina o conteúdo de uma ou mais janelas onde vão ser apresentados os *stats* dos monstros.

PROCWHO

```
3430 DEFine PROCedure who
3440 REPeat inloop
3450 REPeat gloop
3460 g$=INKEY$
3470 IF CODE(g$)>32 THEN EXIT gloop
3480 END REPeat gloop
3490 g$=change$(g$)
3500 FOR j=1 TO LEN(char$)
3510 IF g$=character$(char$(j),1) THE
N ch=char$(j):EXIT inloop
3520 END FOR j
3530 END REPeat inloop
3540 END DEFine who
```

Comentário sobre «Procwho»

Esta rotina é chamada a partir de várias outras. Pegando no *input* que se segue à pergunta «*Who?*», esta rotina atribui à variável *ch* o valor encontrado em *character$*. Na linha 4220, em *Procattack,* encontra-se um exemplo deste trabalho: nesta linha, *ch* tem um valor com o qual a rotina pode trabalhar.

PROCSPELL

```
4990 DEFine PROCedure spell(spels)
5000 PRINT #14,"tries to cast a spell
":PAUSE 100
5010 IF spels=0 THEN PRINT #14,"But h
as no magic":PAUSE 100:RETurn
5020 hit=RND(9)
5030 IF hit=1 THEN PRINT #14,"a miss
":PAUSE 100:RETurn
5040   hit = RND(1 TO 6) MOD 4
5050 PRINT #14,"A hit but only of pow
er ";hit:PAUSE 100
5060 END DEFine spell
```


Comentário sobre «Procspell»

Esta rotina pega no valor transferido para o parâmetro *spels* a partir da linha 4450, em *Procattack,* onde foi lido o terceiro elemento de *character%* (é esse elemento que contém informação sobre os feitiços de que cada personagem dispõe). Se o valor for 0, é claro que não há magia nenhuma à disposição da personagem; se não for 0, é atribuído a *hit* um valor ao acaso, e o resultado do lançamento do feitiço é impresso.

PROCSTRIKE

```
5400 DEFine PROCedure strike(ak,dc)
5410 IF ak>0 THEN
5420 av=RND(1 TO ak)
5430 ELSE
5440 av=0
5450 END IF
5460 IF dc>0 THEN
5470 dv=RND(1 TO dc)
5480 ELSE
5490 dv=0
5500 END IF
5510 IF av>dv THEN
5520 PRINT #14,"A hit":PAUSE 100
5530 hit =1+((RND(av-dv))DIV 2)
5540 RETurn
5550 ELSE
5560 PRINT#14, "The blow misses":PAUS
E 100
5570 END IF
5580 END DEFine strike
```

Comentário sobre «Procstrike»

Esta pequena rotina é o centro da rontina de combate e decide se um golpe atinge o objectivo falha. Esta decisão depende pouco da sorte, estando largamente dependente das capacidades de ataque e defesa dos monstros e personagens, cujo valor é transmitido a esta rotina a partir da linha 4790, em *Procattack*.

PROCFRENZY

```
5070 DEFine PROCedure frenzy
5080 charatt=charatt + (character%(ch
,4) DIV 2)
5090 chardef=chardef-(character%(ch,4
) DIV 3)
5100 END DEFine frenzy
```

PROCCAUTION

```
5110 DEFine PROCedure caution
5120 charatt=charatt-(character%(ch,5
) DIV 3)
5130 chardef=chardef+(character%(ch,5
) DIV 2)
5140 END DEFine caution
```

Comentário sobre «Procfrenzy» e «Proccaution»

Estas rotinas são como que o verso e o reverso da mesma medalha. Em *Procattack*, nas *linhas 4250-4260*, foram dados valores às variáveis *charatt* e *chardef*, sendo estes valores usados nestas duas rotinas para encontrar o resultado do ataque furioso ou cauteloso da personagem.

PROCCHECKCURS

```
5200 DEFine PROCedure chekcurs
5210 kflag=0
5220 FOR kur=6 TO 8
5230 IF character%(ch,kur)<>0 THEN IF
 object%(character%(ch,kur),4)=1 THEN
 charatt=charatt-5
5240 kl= object%(character%(ch,kur),4
)
5250 SELect ON kl
5260 ON kl=1
5270 charatt=charatt-5
5280 ON kl=2
5290 kflag=1
5300 END SELect
5310 END IF
5320 NEXT kur
5330 END DEFine chekcurs
```

Comentário sobre «Proccheckcurs»

Esta rotina é chamada na linha 4540, em *Procattack*, no caso de a personagem atacante transportar um objecto que esteja amaldiçoado. O resultado, nos valores da «bandeira» *kflag* e da variável *charatt*, é passado de novo para *Procattack*.

PROCMONSTERHURT

```
5150 DEFine PROCedure monsterhurt(val
)
5160 r=RND(1 TO 2)
5170 tempmon%(mon,r)=tempmon%(mon,r)-
hit
5180 monup mon
5190 END DEFine monsterhurt
```

Comentário sobre «Procmonsterhurt»

Esta rotina é chamada em várias alturas para actualizar os *stats* de um monstro depois de ter sido registado um golpe que o atingiu.

PROCCHARACTERHURT

```
5340 DEFine PROCedure characterhurt(v
al)
5350 r=RND(1 TO 3)
5370 character%(ch,r)=character%(ch,r
)-hit
5380 char_check tchar$
5390 END DEFine characterhurt
```

Comentário sobre «Proccharacterhurt»

É semelhante a *Procmonsterhurt»,* mas actualiza os *stats* de uma personagem.

PROCTAKE

```
5840 DEFine PROCedure take
5850 CLS#9
5860 IF map%(v,h,2)=0 THEN PRINT#9,"N
othing to take":RETurn
5870 IF map%(v,h,4)<>0 THEN PRINT #9,
"First you'll have to get around the
";:plural:RETurn
5880 PRINT #9,"Who wants to take it?"
5890 who
5900 FOR j=8 TO 6 STEP -1
5910 IF character%(ch,j)=0 THEN takep
=ch:takeh=j
```



```
5920 END FOR j
5930 IF takep=0 THEN PRINT#9,characte
r$(takep);" cannot carry any more":RE
Turn
5940 character%(takep,takeh)=map%(v,h
,2)
5950 map%(v,h,2)=0
5960 cell_display
5970 char_check char$
5980 END DEFine take
```

Comentário sobre «Proctake»

Esta rotina é chamada pelas linhas 380-390, no ciclo principal, se o jogador decidir agarrar um objecto no local em questão. A rotina verifica antes de mais se de facto existe no local alguma coisa que pode ser agarrada. Se não houver, o programa regressa ao menu principal. Porém, se estiver presente um objecto, a *linha 5870* verifica se também se encontra presente um monstro. Em caso afirmativo, o jogador só pode apoderar-se do objecto através de um combate, e o programa regressa ao menu principal, de onde tinha vindo. Note-se que, no fim desta linha, é chamado *Procplural,* de que vamos falar daqui a pouco.

Linhas 5880-5980: estas linhas lêem rapidamente os *stats* das três personagens, com referência particular aos objectos que nessa altura estão a transportar, e, no caso de uma ou mais das personagens poderem transportar ainda mais objectos, distribui entre elas o objecto ou objectos apanhados (no caso de nenhuma poder transportar mais objectos, voltamos ao menu principal).

PROCDRINK

```
2300 DEFine PROCedure drink
2310 CLS#9
2315 CLS#10
2320 sflag=0
2330 FOR i= 1 TO LEN(char$)
2340 char_search char$(i),35
```

```
2350 char_search char$(i),36
2360 NEXT i
2370 IF pl<>1 AND sflag=0 THEN PRINT
#9,"There's no drink here":RETurn
2380 PRINT#9, "Who wants a drink?"
2390 REPeat inloop
2400 person$=INKEY$
2410 person$=change$(person$)
2430 FOR k=1 TO LEN(char$)
2440 IF person$=character$(char$(k),1
) THEN drinker= char$(k):EXIT inloop
2450 NEXT k
2460 END REPeat inloop
2470 IF pl=1 THEN
2480 pl=0
2490 rp=RND(1 TO 2):sp=RND(1 TO 5)
2500 PRINT#9,character$(drinker);"'s
";category$(sp);" is ";
2510 IF rp=1 THEN
2520 PRINT #9,"increased"
2530 character%(drinker,sp)=character
%(drinker,sp)+1
2540 ELSE
2550 PRINT #9,"decreased"
2560 character%(drinker,sp)=character
%(drinker,sp)-1
2570 END IF
2575 char_check char$
2580 RETurn
2590 ELSE
2600 sflag=0
2610 char_search drinker,35
2620 char_search drinker,36
2630 IF sflag<>0 THEN
2640 FOR i=1 TO 5
2650 character%(drinker,i)=character%
(drinker,i)+object%(sflag,i)
2660 END FOR i
2665 PRINT #9,"That feels better"
```



```
2680 character%(drinker,sflag)=0
2690 char_check char$
2700 RETurn
2710 ELSE
2720 FOR i= 1 TO 3
2730 sflag=0
2740 char_search i,35
2750 char_search i,36
2760 IF sflag<>0 THEN
2762 owner=char$(i)
2763 obn=sflag
2765 END IF
2770 END FOR i
2780 rp=RND(10-(character%(owner,5) M
OD 10))
2790 IF rp<2 THEN
2800 PRINT #9,character$(owner);" say
s 'You will have to give me something
 first"
2810 IF character%(drinker,6)=0 THEN
PRINT #9,"but you have nothing I want
":RETurn
2820 ELSE
2830 obmove=character%(owner,obn)
2840 character%(owner,obn)=character%
(drinker,6)
2850 character%(drinker,6)=obmove
2920 PRINT #10,"That'll do nicely"
2930 FOR i=1 TO 5
2940 character%(drinker,i)=character%
(drinker,i)+object%(character%(drinke
r,6),i)
2950 NEXT i
2960 character%(drinker,6)=0
2970 END IF
2980 END IF
2990 char_check char$
3000 CLS#9
3010 END DEFine drink
```

Comentário sobre «Procdrink»

Esta rotina, chamada pelas linhas 400-410, no ciclo principal, permite que as personagens bebam. Podem beber o conteúdo de uma garrafa (que transportam), uma poção (transportada também por um dos membros do grupo), dessedentar-se a partir de um Charco Mágico. Antes de mais nada, uma «bandeira», *sflag,* é colocada a 0, sendo este valor transmitido a *Procchar_search,* que olha para cada personagem para ver se, nesta altura, transporta uma garrafa que pode ser enchida no charco ou uma poção que pode ser bebida. Se a «bandeira» continuar a 0 após esta rotina, isto quer dizer que a personagem não tem nem poção nem garrafa, e que, por conseguinte, não pode beber.

Em *Procpool* é atribuído a *pl* o valor 1 desde que esteja presente um Charco Mágico: a *linha 2370* assegura a existência efectiva de um charco no local; caso este não exista, o controlo passa para o menu principal.

Linhas 2380-2460: Estas linhas actualizam *character$* de acordo com a escolha da personagem que vai beber, escolha essa que é feita pelo jogador.

Linhas 2470-2580: Antes de mais nada estas linhas atribuem à variável do Charco Mágico, *pl,* o valor 0; seguidamente aumentam ou diminuem um atributo da personagem, escolhido ao acaso.

Linhas 2590-2710: Permitem a uma personagem beber a partir da sua própria garrafa, actualizando *character %* consoante for necessário.

Linhas 2720-2980: As restantes linhas tratam do caso em que uma personagem que quer beber tem de aceitar uma bebida oferecida por outro membro do grupo. Nesta situação, a personagem que bebe deve dar a esse membro qualquer coisa em troca. Mas pode não ter sorte! Aconteça o que acontecer, *character %* é sempre mantida actualizada. Finalmente é chamado *proccharacter_check,* que actualiza as janelas de *stats.*

PROCCHAR_SEARCH

```
5790 DEFine PROCedure char_search(c%,
value)
5800 FOR j=6 TO 8
5810 IF character%(c%,j)=value THEN s
flag=j
5820 NEXT j
5830 END DEFine char_search
```

Comentário sobre «Procchar_search»

Esta rotina procura em *character %* o valor que lhe foi transmitido por outra rotina (neste caso, por algumas linhas de *Procdrink*).

PROCPLURAL

```
5660 DEFine PROCedure plural
5670 PRINT #9,monster$(map%(v,h,3));
5680 IF map%(v,h,4)>1 THEN
5690 PRINT#9, "s"
5700 RETurn
5710 ELSE
5720 PRINT#9, " "
5730 END DEFine plural
```

Comentário sobre «Procplural»

Se o número de monstros que se encontram no local for maior que 1, é acrescentado «s» à descrição impressa.

PROCPOOL

```
5740 DEFine PROCedure pool
5750 CLS#9
5760 PRINT#9,"here is a magical pool"
5770 pl=1
5780 END DEFine pool
```

Comentário sobre «Procpool»

Imprime a informação de que existe um Charco Mágico no local de que se está a tratar, e atribui à «bandeira» *pl* o valor 1, que vai ser usado em outras rotinas.

PROCSCORE

```
6360 DEFine PROCedure score
6370 sc=0
6380 FOR i=1 TO LEN(char$)
6390 FOR j=6 TO 8
6400 IF character%(char$(i),j)<>0 THE
N sc=sc+1
6410 END FOR j
6420 END FOR i
6430 sc=(monsterskilled*(turns DIV 6)
)+sc
6440 CLS#9
6450 PRINT #9;"Score so far :";sc
6460 my_word
6470 END DEFine score
```

Comentário sobre «Procscore»

Esta rotina é chamada pelas linhas 420-430, no ciclo principal. É feita a computação da pontuação baseada no número de vezes em que o jogador conseguiu sobreviver, nos monstros que foram mortos e nos objectos encontrados (a informação sobre este último elemento encontra-se em *character %*). No fim deste procedimento é chamada *Procmy_word.*

PROCMY_WORD

```
8430 DEFine PROCedure my_word
8440 spflag=2
8450 r=RND(6)
8460 IF r INSTR char$=0 THEN RETurn
8470 CLS#10
8480 PRINT #10,character$(r);" says:
";
8490 s=RND(3)+1
8500 SELect ON s
8510 =1
8520 grab r
8530 =2
8540 PRINT #10,"I want to rest"
8550 rest
8560 RETurn
8570 =3
8580 PRINT #10,"I'm thirsty"
8590 drink
8600 RETurn
8610 ON s =REMAINDER
8620 IF character%(r,6)=0 THEN grab r
:RETurn
8630 PRINT #10,"I'm fed up with carry
ing this"
8640 drop r
8650 RETurn
8660 END SELect
8670 END DEFine my_word
```

Comentário sobre «Procmy_word»

Esta rotina é chamada ocasionalmente ao longo do programa e dá a impressão de que as personagens falam umas com as outras. De facto, o que acontece é que é feita uma selecção aleatória a partir de um conjunto de possibilidades, entre as quais se encontra a

inexistência de acção, acaso em que parecerá que não se está a passar nada).

Se for gerado o número 1, é chamada *Procgrab,* e a personagem agarra num objecto ou diz alguma coisa. Se for gerado 2, é chamado *Procrest.* Se for chamado 3, é chamada *Procdrink.* No caso de serem gerados outros números é verificado o parâmetro *Nerves* da personagem: se o valor for 0, é-lhe tirada alguma coisa, ou acontece-lhe algo. Se o valor de *Nerves* não for nulo, a personagem farta-se de carregar um objecto qualquer e larga-o.

PROCGRAB

```
8680 DEFine PROCedure grab(a)
8690 IF character%(a,8)<>0 THEN ransa
y:RETurn
8700 b=0
8710 FOR kt= 1 TO LEN(char$)
8720 IF character%(char$(kt),6)<>0 AN
D k<>r THEN b=char$(kt)
8730 NEXT kt
8740 IF b=0 THEN ransay:RETurn
8750 character%(a,8)=character%(b,8):
character%(b,6)=0
8760 PRINT #10,"I'm taking the ";obje
ct$(character%(b,6))
8770 END DEFine grab
```

Comentário sobre «Procgrab»

Esta rotina permite que uma personagem «fane» um objecto pertencente a outra.

PROCREST

```
3020 DEFine PROCedure rest
3030 CLS#9
3040 pl=0
3050 PRINT #9,"You sit down to rest"
3060 s=RND (1 TO LEN(char$)):r=RND(1
TO 2)
3070 PRINT #9;character$(char$(s));"
feels stronger":PAUSE 100
3080 rt=RND(1 TO 3):character%(char$(
s),r)=character%(char$(s),r)+rt
3090 IF map%(v,h,4)<>0 THEN
3100 PRINT #9,"but you have forgotten
 the ";:plural
3110 PAUSE 100:combat
3120 RETurn
3130 ELSE
3140 re=RND(1 TO no_of_events)
3150 SELect ON re
3160 =1:cursmove
3170 =2:pool
3180 =3:voice
3190 =4:randomonster
3200 END SELect
3205 END IF
3210 char_check char$
3220 END DEFine rest
```

Comentário sobre «Procrest»

Esta rotina, chamada pelas linhas 450-460, no ciclo principal, verifica se, num determinado local, existe algum monstro e, se necessário (isto é, se existir mais de um), chama *Procplural.* Em caso afirmativo o programa vai para *Proccombat,* cuja acção o jogador já conhece. Se não houver monstro nenhum, a rotina não acaba logo: é gerado um número aleatório (usando a variável *no-of-events,*

à qual é inicialmente atribuído o valor 4, na linha 7060, em *Procinit*) de acordo com o qual são chamadas várias rotinas. Finalmente, *Procchar_check* actualiza a apresentação de cada personagem.

As quatro possibilidades de rotinas que temos aqui são bastante divertidas e, ainda que o Charco Mágico tenha tido uma utilidade prévia neste programa, as outras rotinas ocorreram-nos independentemente da sua utilidade, e apenas enquanto estávamos a concretizar o programa. São rotinas bastante fáceis de seguir e mostram como um bocado de pensamento e imaginação suplementares podem acrescentar imensa atmosfera ao programa.

PROCCURSMOVE

```
3300 DEFine PROCedure cursmove
3310 FOR j=1 TO LEN(char$)
3320 FOR k= 6 TO 8
3330 IF object%(character%(char$(j),k
),4)=3 THEN
3340 REPeat hloop
3350 s=RND(1 TO 24)
3360 r=RND(1 TO 12)
3370 IF map%(r,s,1)<>0 THEN v=r:h=s:E
XIT hloop
3380 END REPeat hloop
3390 END IF
3400 NEXT k
3410 NEXT j
3420 END DEFine cursmove
```

Comentários sobre «Proccursmove»

Esta é a primeira das novas rotinas. Se uma das personagens estiver a transportar um objecto amaldiçoado *(linha 3330)*, todo o grupo é transportado para um local ao acaso — uma partida muito desagradável! Entre outras coisas, isto torna mais difícil ao jogador fazer o mapa do conjunto de locais e destrói qualquer estratégia que ele tenha planeado.

PROCVOICE

```
9160 DEFine PROCedure voice
9165 CLS#10
9170 CLS#9
9180 PRINT#9;"A distant voice says: "
;
9190 qr=RND(1 TO 4)
9200 SELect ON qr
9210 =1:ransay
9220 =2:distance 2
9230 =3:distance 3
9240 =4:cursay
9250 END SELect
9260 END DEFine voice
```

Comentário sobre «Procvoice»

Esta é a segunda das novas rotinas, e tem um efeito que compensa o de *Proccursmove.* Ouve-se uma voz distante, de origem indefinida, que dá ao grupo alguns conselhos. Aqui há quatro possibilidades (em três rotinas: *Procdistance, Proccursay* e *Procrandomonster*), dependentes do valor aleatório atribuído a *gr* na *linha 9190.*

PROCDISTANCE

```
9340 DEFine PROCedure distance(item)
9350 find=0:k=1
9360 REPeat bloop
9365 j=1
9370 REPeat cloop
9380 IF map%(j,k,item)<>0 AND (j<>v O
R k<>h) THEN find=ABS((ABS(v-j))-(ABS
(h-k)))
9390 IF j>11 OR find<>0 THEN EXIT clo
op
```

```
9400 j=j+1
9410 END REPeat cloop
9420 IF k>23 OR find <>0 THEN EXIT bl
oop
9430 k=k+1
9440 END REPeat bloop
9450 IF find=0 THEN RETurn
9460 SELect ON item
9470 =3 :PRINT #9,"A monster is ";fin
d;" rooms away"
9480 =2 :PRINT #9,"A ";object$(map%(j
,k,2));" is ";find;" rooms away"
9485 END SELect
9490 END DEFine distance
```

Comentário sobre «Procdistance»

Se o valor atribuído a *gr* for 2 ou 3, é chamada esta rotina, que informa o grupo da localização de monstros ou de objectos. O valor de *gr* decide se se vai tratar de um monstro ou de um objecto.

PROCCURSAY

```
9270 DEFine PROCedure cursay
9280 FOR i=1 TO LEN(char$)
9290 FOR j=6 TO 8
9300 IF object%(character%(char$(i),j
),6)<>0 THEN PRINT #9,character$(char
$(i));" is cursed"
9310 END FOR j
9320 END FOR i
9325 PRINT #9,"Be careful"
9330 END DEFine cursay
```

Comentário sobre «Proccursay»

Se esta rotina for chamada, a voz avisa o grupo no caso de uma das personagens estar a transportar um objecto amaldiçoado.

De novo em *Procrest*, vamos tratar do quarto acontecimento aleatório *(linha 3190)*. Vai ser outro acontecimento desagradável, para manter o grupo apreensivo.

PROCRANDOMONSTER

```
3230 DEFine PROCedure randomonster
3240 s=RND(1 TO 16)
3250 map%(v,h,4)=1:map%(v,h,3)=s
3260 CLS#14:PRINT #14," Suddenly a ";
monster$(s);" appears":PAUSE 100
3270 cell_display
3280 combat
3290 END DEFine randomonster
```

Comentário sobre «Procrandomonster»

Esta rotina chama um monstro ao acaso, actualiza a apresentação da cela e dá início à sequência de batalha.

PROCTALK

```
5590 DEFine PROCedure talk
5600 CLS#9
5610 IF map%(v,h,4)=0 THEN PRINT #9;"
Your chatter might attract a monster"
:RETurn
5620 spflag=1
5630 PRINT#9,"A ";monster$(map%(v,h,3
));" says: ";
5640 ransay
5650 END DEFine talk
```

Comentário sobre «Proctalk»

Esta rotina, que é muito curta mas bastante importante, é chamada pelas linhas 460-470, no ciclo principal. É chamada quando o jogador quer falar a um monstro, e chama por sua vez uma rotina maior, *Procransay*. Esta rotina também é usada pelas personagens, de forma que algumas das coisas que diz estão um bocado deslocadas! Se não houver monstro nenhum no local, o programa volta simplesmente ao menu principal.

PROCRANSAY

```
8910 DEFine PROCedure ransay
8920 r=RND(3)
8930 SELect ON r
8940 =1
8950 bita=RND(5)+1:bitb=RND(5)+1
8960 PRINT #(8+spflag);first$(bita);"
 ";second$(bitb)
8970 ON r=REMAINDER
8980 PRINT #(8+spflag),"I'd like to "
;nasty$(RND(5)+1);
8990 IF spflag=2 THEN
9000 PRINT #(8+spflag)," a ";monster$
(RND(15)+1)
9010 ELSE
9020 PRINT #(8+spflag)," an adventure
r"
9030 END IF
9040 =2
9050 PRINT #(8+spflag),"Where's the "
;object$(RND(35)+1);" then?"
9060 END SELect
9070 END DEFine ransay
```


Comentário sobre «Procransay»

Esta rotina (*RANdom SA Yings,* dizeres ao acaso) é chamada a partir de um par de sítios do programa. Para assegurar que não seja dito nada de *demasiado* pouco natural — ou seja, para que os monstros digam coisas mais ou menos monstruosas, enquanto as personagens dizem coisas mais ou menos humanas —, a «bandeira» *spflag* é colocada em 1 quando a rotina é chamada a partir de *Procvoice* e em 2 quando é chamada a partir de *Proctalk*.

PROCGIVE

```
5990 DEFine PROCedure give
6000 CLS#14
6010 IF map%(v,h,4)=0 THEN CLS#9:PRIN
T #9,"There's no-one here to take it"
:RETurn
6020 PRINT #14,"What are you going to
 give?"
6030 INPUT#14, gift$
6040 FOR i=1 TO LEN(gift$)
6050 IF CODE(gift$(i))<97 THEN gift$(
i)=CHR$(CODE(gift$(i))+32)
6060 NEXT i
6070 gno=0
6080 FOR i=1 TO 36
6090 IF object$(i,2)=gift$ THEN gno=i
6100 END FOR i
6110 IF gno=0 THEN PRINT#9,"No such o
bject":RETurn
6115 okfl=0
6120 FOR i=1 TO LEN(char$)
6130 FOR j= 6 TO 8
6140 IF character%(char$(i),j)=gno TH
EN okfl=1
6145 END FOR j
6150 END FOR i
```

```
6155 IF okfl=0 THEN RETurn
6160 IF bargain=1 THEN bargain=0:acce
pt
6170 IF RND(1 TO 9)<(monster%(map%(v,
h,3),5)) THEN accept
6180 char_check char$
6190 RETurn
6200 END IF
6230 PRINT #9;"No-one has the";object
$(gno)
6240 END DEFine give
```

Comentário sobre «Procgive»

Esta rotina é chamada pelas linhas 480-490, no ciclo principal, e permite que o jogador ofereça um presente a um monstro. Após ter verificado a existência de um ser a quem a oferta possa ser feita, a rotina verifica também a existência do presente que o jogador quer oferecer e assegura-se de que o mesmo é transportado por um dos membros do grupo. Se o valor da variável *bargain* for 1, o monstro vai aceitar o presente (ver *Procaccept*); se for 0 (esta variável é inicialmente posta a 0 na linha 7000, *Procinit*), há uma probabilidade aleatoriamente determinada de que o monstro o aceite na mesma. Finalmente, *Procchar_check* é chamado, na *linha 7180,* para actualizar o conteúdo do visor.

PROCACCEPT

```
6250 DEFine PROCedure accept
6260 map%(v,h,3)=0:map%(v,h,4)=0
6261 FOR i=1 TO LEN(char$)
6262 FOR j=6 TO 8
6263 IF character%(char$(i),j)=gno TH
EN
6264 character%(char$(i),j)=0
6265 END IF
6266 END FOR j
```



```
6267 END FOR i
6270 PRINT #9;"Thanks for the gift, y
ou can pass"
6275 cell_display
6280 END DEFine accept
```

Comentário sobre «Procaccept»

Esta rotina actualiza *map%* e imprime informação sobre a aceitação do presente pelo monstro.

De novo no ciclo principal, a linha 500 inicia as rotinas de saída, que incluem a opção de salvaguardar o jogo conjuntamente com todos os dados.

No entanto, antes disto, o programa verifica se as várias condições necessárias para a vitória foram cumpridas. Seja como for, saímos agora do ciclo principal. As linhas 530 e 540 lêem *object%* para ver se alguma das personagens tem em seu poder o objecto da demanda (como foi definido pelo utilizador em QLAG). Se assim for, é atribuído a *vic* o valor 1, sendo chamada seguidamente *Proccelebrate*. Mas se todas as personagens tiverem morrido (ou seja, se *char$* = 0) é chamada *Proccry.*

PROCCRY

```
6290 DEFine PROCedure cry
6295 FOR j= 1 TO 8
6300 FOR i =8 TO 13
6310 PAPER#(i),0
6315 INK #(i),i-7
6320 CLS#(i)
6325 PAUSE ((8-j)*2)
6330 PRINT #(i),"SHAME!!"
6340 NEXT i
6345 NEXT j
6350 END DEFine cry
```

Comentário sobre «Proccry»

Uma exibição colorida anuncia a trágica morte do grupo de personagens!

Segue-se o fim do programa.

PROCCELEBRATE

```
6570 DEFine PROCedure celebrate
6580 FOR j=1 TO 10
6585 FOR i=8 TO 13
6590 CLS#(i)
6595 PAPER #(i),i-7
6600 PAUSE 10
6610 PRINT #(i),"Congratulations"
6620 NEXT i
6630 NEXT j
6640 END DEFine celebrate
```

Comentário sobre «Proccelebrate»

Aparece de novo uma exibição colorida, mas desta vez celebrando a vitória.

No entanto, o programa pode chegar à conclusão de que não se deu nenhum desses casos, sendo dada ao jogador a hipótese de salvaguardar todos os dados e a posição do grupo de personagens.

PROCFILE_OUT

```
7870 DEFine PROCedure file_out
7880 PRINT #14,"Please type name of g
ame"
7890 INPUT #14,game$
7900 PRINT #14,"Place the cartridge i
```



```
nto Microdrive 1"
7910 PRINT #14,"And press <Enter>"
7920 REPeat get_loop
7930 in$=INKEY$
7940 IF in$=CHR$(10) THEN EXIT get_lo
op
7950 END REPeat get_loop
7960 gn$="MDV1_"& game$ & "numbers"
7970 OPEN_NEW #15, gn$
7980 FOR i=1 TO 36
7990 FOR j=1 TO 6
8000 PRINT #15,object%(i,j)
8010 NEXT j
8020 NEXT i
8030 FOR i=1 TO 12
8040 FOR j=1 TO 24
8050 FOR k=1 TO 4
8060 PRINT #15,map%(i,j,k)
8070 NEXT k
8080 NEXT j
8090 NEXT i
8100 FOR i=1 TO 16
8110 FOR j=1 TO 6
8120 PRINT #15,monster%(i,j)
8130 NEXT j
8140 NEXT i
8150 FOR i=1 TO 6
8160 FOR j=1 TO 5
8170 PRINT #15,character%(i,j)
8180 NEXT j
8190 NEXT i
8200 PRINT#15,turns
8210 PRINT#15,monsterskilled
8220 PRINT#15,v
8230 PRINT#15,h
8240 CLOSE #15
8250 gm$="mdv1_"&game$&"names"
8260 OPEN_NEW #15, gm$
8270 FOR i= 1 TO 12
8280 FOR j=1 TO 24
```

```
8290 PRINT #15,map$(i,j)
8300 NEXT j
8310 NEXT i
8320 FOR i= 1 TO 36
8330 FOR j=1 TO 3
8340 PRINT #15,object$(i,j)
8350 NEXT j
8360 NEXT i
8370 FOR i =1 TO 6
8380 PRINT #15,character$(i)
8390 NEXT i
8400 PRINT#15,char$
8410 CLOSE #15
8420 END DEFine file_out
```

Comentário sobre «Procfile_out»

Como vimos há já algum tempo, quando discutíamos a rotina *Procinfile,* também esta rotina pode ser transportada a partir do programa QLAG (o nosso gerador), previamente teclado. Seguindo as instruções que se encontram no comentário a *Procinfile,* o utilizador pode poupar uma carga de trabalho salvaguardando e anexando a rotina semelhante pertencente a QLAG. Não se esqueça, no entanto, de mudar os números das janelas consoante for necessário. O nome da rotina em QLAD é o mesmo, e pode ser encontrada nas linhas 9000-9500 desse programa.

Chegámos finalmente ao fim do programa. Introduzi-lo através do teclado foi, pelo menos até aqui, um passatempo bastante agradável e terapêutico (pelo menos esperamos que sim!) Contudo, chegamos agora a um trabalho muito maçador. Os dados a serem utilizados pelas rotinas de imagens da aventura são, infelizmente, muito longos, e o trabalho de tecleio vai revelar-se uma tarefa árdua. Como já sugerimos no início do programa, não é absolutamente necessário passar por toda esta trabalheira: o jogador pode, muito compreensivelmente, optar por uma aventura em texto. A estrutura de QLAG pode perfeitamente ser usada para esse fim e, nesse caso, estas linhas de dados não são necessárias. No entanto,

se decidir seguir o caminho das imagens, vai precisar de ginasticar os dedos e de se atirar ao teclado determinadamente.

Há duas formas de aliviar o terror que estas linhas causam. A primeira é ler o capítulo seguinte, onde vai encontrar um programa que lhe permite planear as suas próprias imagens: os dados de que irá necessitar vão ser escritos por si, e teclar os seus próprios dados deve ser, a longo prazo, mais compensador. A segunda forma baseia-se no facto de não precisar de copiar todas as linhas dos nossos dados para ir para a frente: se quiser apenas ver qual é o aspecto da aventura, ou se estiver indeciso entre uma aventura só em texto ou com imagens, apenas tem de usar o QLAG para descrever um par de monstros e introduzir os dados necessários. Se for este o caso, não se esqueça de DIMensionar *monster%,* na linha 12080 de QLAG com, por exemplo, (3,6), se achar que é suficiente e, como é óbvio, voltar a DIMensioná-la subsequentemente! Salvaguarde estes dados, passe o programa QAD e jogue o seu jogo de aventuras! Se gostar do resultado, pode introduzir o resto dos dados.

Se quiser excluir completamente as imagens (inclusive a representação gráfica de cada divisão) tem de retirar as rotinas *Procmon_show, Procobj_show, Procdoor_draw* e *Procgraphic,* bem como as respectivas chamadas para essas rotinas. Dado que são rotinas referentes a imagens, são desnecessárias para uma aventura em texto.

DECLARAÇÕES DE DADOS

```
9500 DATA "I'd like to","go home","st
rangle","Let's ","kill something","an
nihilate","Why don't we","have a drin
k","torment"
9510 DATA "I think you should","have
a rest","make a stew from","Its time
to","steal some treasure","beat up","
I want to","retire","roast"
9520 DATA "snake","ghost","wild dog",
"roper","gnome","dragon","spider","tr
oll","ogre","wizard","hulk","ghoul","
```

```
ostrich","vulture","demon","lionman"
9530 DATA "Attack ","Defence","Spells
 ","Anger  ","Nerves","Greed  "
10000 REMark monster graphic data
10010 REMark snake
10020 DATA 18,2
10030 DATA 42,128
10040 DATA 554,168
10050 DATA 170,0
10060 DATA 162,168
10070 DATA 160,0
10080 DATA 168,0
10090 DATA 40,0
10100 DATA 40,0
10110 DATA 42,0
10120 DATA 10,128
10130 DATA 10,160
10140 DATA 0,160
10150 DATA 32,168
10160 DATA 128,40
10170 DATA 128,40
10180 DATA 160,40
10190 DATA 170,168
10200 DATA 42,160
10210 DATA 2,128
10320 REMark ghost
10330 DATA 18,4
10340 DATA 0,10815,0
10350 DATA 43260,43775,35535
10360 DATA 43260,35054,35335
10370 DATA 43260,35054,35335
10380 DATA 10300,43775,35335
10390 DATA 10300,43775,35335
10400 DATA 10300,43775,35335
10410 DATA 2575,43775,43775
10420 DATA 2575,43775,43775
10430 DATA 515,43775,41200
10440 DATA 0,43775,32960
10450 DATA 0,43775,32960
```



```
10460 DATA 0,43775,32960
10470 DATA 515,43775,41200
10480 DATA 515,43775,41200
10490 DATA 2575,43775,43260
10500 DATA 2575,43775,43260
10510 DATA 10815,43775,43775
10520 DATA 10815,43775,43775
10610 REMark wild dog
10620 DATA 7,4
10630 DATA 10815,33475,0
10640 DATA 8240,515,41200
10650 DATA 10815,43775,10943
10660 DATA 10815,43775,41200
10670 DATA 10300,10300,10815
10680 DATA 43260,2575,0
10690 DATA 32960,0,32960
10700 DATA 32960,0,32960
10910 REMark roper
10920 DATA 9,4
10930 DATA 40960,0,40960
10940 DATA 2048,2560,0
10950 DATA 35344,2048,8192
10960 DATA 33280,33280,10752
10970 DATA 40960,33280,512
10980 DATA 10752,43520,41476
10990 DATA 2050,10370,8836
11000 DATA 2560,170,43520
11010 DATA 10752,33320,40960
11020 DATA 43520,43520,43520
11210 REMark gnome
11220 DATA 10,4
11230 DATA 0,0,2575
11240 DATA 10943,43775,10943
11250 DATA 192,3,2767
11260 DATA 202,168,192
11270 DATA 714,8360,192
11280 DATA 2254,2220,192
11290 DATA 2815,43263,192
11300 DATA 2815,10431,192
```

```
11310 DATA 575,41215,0
11320 DATA 10752,2800,0
11330 DATA 43520,2560,32768
11500 REMark dragon
11520 DATA 28,6
11530 DATA 0,0,0,2
11540 DATA 0,0,12,136
11550 DATA 0,0,575,160
11560 DATA 0,0,60,168
11570 DATA 0,0,60,2060
11580 DATA 0,3,255,252
11590 DATA 0,63,191,240
11600 DATA 0,3,255,0
11610 DATA 0,255,240,32896
11620 DATA 2056,3,240,2248
11630 DATA 117,15,192,240
11640 DATA 562,15,192,192
11650 DATA 60,63,3,192
11660 DATA 15,63,15,0
11670 DATA 15,255,255,0
11680 DATA 3,255,240,0
11690 DATA 0,63,0,0
11700 DATA 0,15,0,0
11710 DATA 0,15,192,0
11720 DATA 0,195,240,0
11730 DATA 3,3,252,0
11740 DATA 12,0,255,0
11750 DATA 15,0,63,192
11760 DATA 3,240,15,192
11770 DATA 0,252,3,192
11780 DATA 0,63,3,192
11790 DATA 0,15,255,192,0,3,255,0,0,0
,252,0
11800 REMark spider
11820 DATA 10,4
11830 DATA 10815,0,43260
11840 DATA 8759,0,35036
11850 DATA 10815,0,43260
11860 DATA 2056,0,8224
```



```
11870 DATA 2056,10280,8224
11880 DATA 10794,43690,43176
11890 DATA 43690,170,43690
11900 DATA 43690,43690,43690
11910 DATA 8738,8738,8224
11920 DATA 8738,8738,8224
11930 DATA 8738,8738,8224
12110 REMark troll
12120 DATA 9,4
12130 DATA 0,43008,0
12140 DATA 0,35360,0
12150 DATA 20,43008,0
12160 DATA 21,80,43260
12170 DATA 85,64,43260
12180 DATA 84,32768,32960
12190 DATA 84,10752,32768
12200 DATA 35328,512,32768
12210 DATA 33280,0,32960
12220 DATA 41715,32960,0
12410 REMark ogre
12420 DATA 27,6
12430 DATA 0,10752,32768,0
12440 DATA 512,43520,43008,0
12450 DATA 512,2560,2048,0
12460 DATA 512,43520,43008,0
12470 DATA 2560,32810,10880,0
12480 DATA 2560,10880,35360,0
12490 DATA 2560,43520,43520,0
12500 DATA 40965,598,85,40960
12510 DATA 40965,598,85,40960
12520 DATA 40965,598,85,40960
12530 DATA 40965,598,85,40960
12540 DATA 40965,2650,32917,40960
12550 DATA 40965,43690,43177,40960
12560 DATA 43520,43520,43520,40960
12570 DATA 43520,43520,43520,40960
12580 DATA 43520,43520,43520,40960
12590 DATA 1,2650,32917,0
12600 DATA 1,2650,32917,0
```

```
12610 DATA 1,2650,32917,0
12620 DATA 5,2650,32917,64
12630 DATA 5,2650,32917,64
12640 DATA 5,2650,32917,64
12650 DATA 5,2634,32901,64
12660 DATA 21,578,32901,80
12670 DATA 21,514,32897,80
12680 DATA 21,514,1,80
12690 DATA 43520,514,512,43008
12695 DATA 43520,514,512,43008
12700 REMark wizard
12720 DATA 17,4
12730 DATA 0,16,0
12740 DATA 0,64,2060
12750 DATA 1,64,2060
12760 DATA 1,64,10815
12770 DATA 5,80,10815
12780 DATA 21,84,2060
12790 DATA 522,160,2060
12800 DATA 10,128,2060
12810 DATA 2,128,2060
12820 DATA 1,85,104
12830 DATA 1,69,2124
12840 DATA 5,64,2124
12850 DATA 5,80,2060
12860 DATA 21,80,2060
12870 DATA 21,84,2060
12880 DATA 85,84,2060
12890 DATA 85,85,2060
12900 DATA 85,85,2060
13010 REMark hulk
13020 DATA 12,6
13030 DATA 10240,2560,32768,40960
13040 DATA 43520,10752,41472,43008
13050 DATA 43520,8708,8768,43008
13060 DATA 43520,43520,43520,43008
13070 DATA 43520,41480,10880,43008
13080 DATA 43520,40970,10880,43008
```



```
13090 DATA 43008,43520,43008,43008
13100 DATA 10240,10752,40960,40960
13110 DATA 10240,255,252,40960
13120 DATA 10240,252,252,40960
13130 DATA 43520,252,764,43008
13140 DATA 43520,43176,43688,43008
13150 DATA 43520,43176,43688,43008
13310 REMark ghoul
13320 DATA 25,8
13330 DATA 0,43775,43775,32960,0
13340 DATA 0,33475,41200,32960,0
13350 DATA 0,33475,41200,32960,0
13360 DATA 0,43775,43775,32960,0
13370 DATA 0,10815,43775,0,0
13380 DATA 0,8755,8755,0,0
13390 DATA 0,8755,8755,0,0
13400 DATA 0,10815,43775,0,0
13410 DATA 0,515,41200,0,0
13420 DATA 43260,515,41200,2575,32960
13430 DATA 43775,43775,43775,43775,32
960
13440 DATA 43260,0,32960,2575,32960
13450 DATA 8240,43775,43775,33475,0
13460 DATA 8240,0,32960,515,0
13470 DATA 8240,10815,43775,515,0
13480 DATA 8240,0,32960,515,0
13490 DATA 8240,2575,43260,515,0
13500 DATA 41715,41200,33475,41715,32
960
13510 DATA 41715,43775,43775,41715,32
960
13520 DATA 515,41715,41715,41200,0
13530 DATA 0,32960,32960,32960,0
13540 DATA 0,32960,0,32960,0
13550 DATA 0,32960,0,32960,0
13560 DATA 0,32960,0,32960,0
13570 DATA 0,32960,0,32960,0
13580 DATA 10815,41200,515,43775,0
13590 REMark ostrich
13600 DATA 16,4
```

```
13610 DATA 168,0,0,154,128,0
13620 DATA 170,130,160,40,2,106
13630 DATA 8,2,170,8,2,128
13640 DATA 10,10,0,2,8,0
13650 DATA 2,8,0,2,168,0
13660 DATA 10,170,0,10,170,0
13670 DATA 2,168,0,2,8,0
13680 DATA 2,8,0,2,8,0,10,10,0
13890 REMark vulture
13900 DATA 7,6
13910 DATA 0,10773,41040,0,0,8793,834
0,0
13920 DATA 1,2645,32853,0,5,534,81,64
13930 DATA 20,518,64,80,80,21,80,20
13940 DATA 64,16,16,4,64,68,68,4
14190 REMark demon
14200 DATA 18,6
14210 DATA 64,0,0,16,64,32768,8192,16
14220 DATA 80,32768,8192,80,80,8192,3
2768,80
14230 DATA 84,8197,32769,80,20,21,65,
64
14240 DATA 21,37,133,64,21,21,69,64
14250 DATA 5,5,5,0,5,85,85,0
14260 DATA 1,85,84,0,5,85,84,0
14270 DATA 5,85,85,0,5,21,69,0
14280 DATA 20,21,65,64,20,80,81,64
14290 DATA 80,64,16,80,576,40960,4300
8,16,576,8192,34816,16
14490 REMark lionman
14500 DATA 21,6
14510 DATA 2,682,32896,0,2,10922,4112
0,0
14520 DATA 2,10409,64,0,10,10922,4112
8,0
14530 DATA 10,43690,43176,0,10,43176,
43192,0
14540 DATA 2,43176,8240,0,0,43176,0,0
```



```
14550 DATA 0,43690,0,41120,160,10794,
43690,43196
14560 DATA 160,10794,41120,8224,8224,
10794,0,0
14570 DATA 8224,10794,0,0,8224,43690,
32896,0
14580 DATA 33410,10794,32896,0,33410,
10794,32896,0
14590 DATA 10280,2570,32896,0,0,2570,
32896,0
14600 DATA 0,2570,32896,0,0,2570,0,0
14610 DATA 0,2056,0,0,0,10794,41120,0
14660 REMark Object data
14670 DATA 5,2,0,8240,0,32960,515,0,3
5020,0,8240,0,35020,0
14680 DATA 8,2,0,8224,0,32896,2570,0,
10794,0,43176,0,84,0,84,0,16,0,16,0
14690 DATA 5,2,2,160,10,160,10,160,42
,128,168,0,160,0
14700 DATA 3,2,10260,0,85,85,10260,0,
43605,0
14710 DATA 5,0,2060,10815,2060,2060,2
060,2060
14720 DATA 7,0,10280,33410,33410,514,
514,514,514,514
14730 DATA 6,0,43775,43523,43523,4352
3,43523,43523,43523
14740 DATA 1,0,2575,2575
14750 DATA 6,2,2060,0,10815,0,2060,0,
42,0,170,128,170,128,42,0
14760 DATA 5,0,34884,8208,34884,8192,
8192,8192
14770 DATA 7,0,40,43690,43690,40,40,4
0,40,40
14780 DATA 3,0,33345,10260,10260,3334
5
14790 DATA 3,0,10280,33410,33410,1028
0
14800 DATA 5,2,32768,0,40960,0,10240,
0,10752,0,2560,40960,2560,43008
```

```
14810 DATA 4,2,8224,32896,32896,8224,
32896,8224,8224,32896,2570,0
14820 DATA 5,2,10815,0,43775,32960,32
960,32960,41715,32960,41715,32960,417
15,32960
14830 DATA 3,2,43775,43260,515,43775,
33475,43775,10815,43260
14840 DATA 3,0,34986,8874,34986,8874
14850 DATA 5,2,43775,41200,43775,4120
0,10815,32960,10815,32960,2575,0,2575
,0
14860 DATA 2,0,32,168,40
14870 DATA 2,0,65,85,65
14880 DATA 4,0,80,80,80,85,85
14890 DATA 6,2,43775,35016,32960,8224
,8240,34956,2574,2060,515,2060,0,3502
0,0,10300
14900 DATA 2,0,43690,43710,43690
14910 DATA 6,0,10,8,40,168,168,170,17
0
14920 DATA 2,2,5,0,5,0,85,80
14930 DATA 3,0,8240,43260,8240,8240
14940 DATA 4,0,43260,43260,8240,8240,
43260
14950 DATA 2,2,34952,32896,41642,3289
6,43690,32896
14960 DATA 5,0,8224,43192,8224,8224,8
224,8224,8224
14970 DATA 2,4,0,41200,0,170,41210,17
0,0,41200,0
14980 DATA 4,2,43605,41044,10773,3283
2,10773,32832,10773,32832,10773,32832
14990 DATA 4,2,10,0,8240,32960,32960,
8240,32960,8240,8240,32960
15000 DATA 5,0,515,2575,10300,41200,3
2960,32960
15010 DATA 5,0,8224,43176,43092,43176
,43176,43176
15020 DATA 2,0,10260,43605,43605
```


ATÉ QUE ENFIM!

Quer tenha introduzido os dados ou decidido não o fazer, chegámos ao fim da listagem.

Agora, tudo o que tem a fazer é jogar a aventura. Estamos convictos de que este jogo é não só uma utilidade preciosa para quem quer que esteja interessado neste sector da programação de computadores pessoais, mas também a melhor aventura para o *QL*...

CAPÍTULO 8

Imagens no «QL»

Ao contrário do que acontece com o *Spectrum* e outros microcomputadores de grande venda, o *QL* não tem «gráficos definidos pelo utilizador». Assim, se quisermos criar monstros colorindo *pixels* sobre o visor em vez de desenharmos linhas preenchidas por formas, temos de enviar valores directamente para a memória do visor.

O *QL* tem um visor a que corresponde na memória uma representação do tipo mapa. Isto significa que existe pelo menos uma localização de memória correspondente a cada «localização» do visor. Desta forma, se soubermos qual é o *bit* ou *byte* da RAM (*random access memory,* memória de acesso aleatório) que controla cada parte do visor, podemos alterar os valores que se encontram nessa porção de RAM, modificando dessa forma aquilo que nos é apresentado no visor.

No *QL,* a memória do visor começa em 131072. Se se colocar (POKE) um valor nesta localização, vai aparecer no canto superior direito do seu visor um arranjo de pontos (desde que o seu monitor de TV tenha possibilidade de apresentar todos os pontos que o QL pode enviar). Em baixa resolução (o tipo usado em ambos os programas deste livro, QLAG e QAD) o visor está ordenado em unidades de quatro *pixels.* Não é possível tratar de *menos* de quatro *pixels* de cada vez. Cada conjunto de quatro é controlado por um par de *bytes* a que se chama uma «palavra». Cada palavra está dividida em oito pares de *bits,* contendo um par em cada *byte* a informação respeitante a um *pixel* do visor. O arranjo toma esta forma:

BYTES	*byte* superior				*byte* inferior			
BITS	7/6	5/4	3/2	1/0	7/6	5/4	3/2	1/0
PIXEL	1	2	3	4	1	2	3	4
CORES	F/G	F/G	F/G	F/G	R/B	R/B	R/B	R/B

(F=*flash,* brilho; *G=green,* verde; R=*red,* encarnado; B=*blue,* azul)

Se quisermos que o quarto *pixel* do visor seja encarnado, temos de activar o *bit* 1 do *byte* inferior; se quisermos que o segundo e o terceiro *pixels* sejam verdes, temos de activar os *bits* 4 e 2 do *byte* superior; se quisermos uma cor que não seja uma destas cores básicas, temos de activar dois ou mais *bits* que controlam o mesmo *pixel*. Por exemplo, para que o *pixel* 3 tome a cor vermelho-escura, activamos o *bit* para o encarnado e o *bit* para o azul do *pixel* 3, que são os *bits* 2 e 3 do *byte* inferior. Se quisermos que o *pixel* 1 seja branco, activamos os *bits* para o encarnado, o azul, e o verde desse *pixel,* ou seja, os *bits* 6 e 7 do *byte* inferior e o *bit* 6 do *byte* superior.

Uma vez que cada combinação de *bits* activados e não activados de uma palavra representa um número entre 0 e 65535, há 65535 padrões de cor possíveis (incluindo o brilho) para um grupo de quatro *pixels*. É bastante aborrecido descobrir quais são os *bits* que devem ser activados para cada *pixel* do visor (especialmente devido ao facto de os *pixels* se contarem da esquerda para a direita, enquanto os *bits* se contam da direita para a esquerda. Assim, resolvemos incluir aqui um programa curto e imperfeito, mas que pode fazer por si o trabalho de calcular o número decimal que é codificado por uma determinada combinação de *bits*. Foi este programa que fez esse trabalho por nós quando planeávamos as imagens para o programa de aventuras deste livro!

Desenhe simplesmente as suas imagens em papel milimétrico e divida-as em unidades horizontais de quatro pontos. Para cada unidade faça passar um ciclo do programa (que é um ciclo de repetição infindável). O programa pede a letra inicial da cor de cada um dos quatro *pixels,* referindo-se a estes por meio do *bit* da direita dos *bytes* de cada unidade (para o preto é *x,* pois não é considerado cor). Seguidamente, o programa pede o «*Bit* 1», e o jogador deve responder introduzindo a letra inicial da cor do *pixel* esquerdo da sua unidade. Quando as quatro cores tiverem sido introduzidas, o programa fornece um valor decimal. Se o utilizador colocar

(POKE) este valor no endereço de memória de uma localização do visor, vai ver aparecer magicamente o seu padrão de pontos. O programa não fornece pontos brilhantes, porque fazer um ponto brilhar faz brilhar o resto da linha. As cores que ele fornece são: azul (B=*blue*), encarnado (R=*red*), vermelho-escuro (M=*magenta*), Turquesa (C=*cyan*), amarelo (Y=*yellow*), branco (W=*white*) e verde (G=*green*).

Todos os monstros e objectos incluídos em QAD são desenhados usando códigos derivados deste programa. É trazido um indicador para a primeira linha do bloco de dados respeitantes à imagem requerida. O número de iterações verticais e horizontais é lido para o computador a partir destes dados. Seguidamente, à medida que os grupos de quatro horizontais são multiplicados pela altura da imagem (isto é, pelo número de *pixels* na vertical), os dados são lidos para o computador e colocados (POKE) nos endereços de memória correspondentes ao ponto do visor onde devem ficar.

CRIAÇÃO DE GRÁFICOS

```
100 REMark GRAPHIC SET-UP PROGRAM
110 CLS
120 REPeat main
130 s=0:t=0
140 FOR i=1 TO 7 STEP 2
150 fact=0
160 PRINT "Bit ";i
170 col$=" "
180 REPeat loop
190 cpos=0
200 col$=INKEY$
210 IF col$="" THEN GO TO 200
220 cpos =col$ INSTR "brmxcywg"
230 IF cpos <>0 THEN EXIT loop
240 END REPeat loop
250 lsb=cpos MOD 4
260 IF lsb =1 OR lsb=3 THEN fact=fact
  +(2^(i-1))
```

```
270 IF lsb=2 OR lsb=3 THEN fact=fact+
(2^i)
280 IF cpos>4 THEN
290 msb=2^(i+8)
300 ELSE
310 msb=0
320 END IF
330  t= msb+fact
340 s=s+t
350 NEXT i
360 PRINT "Code= ";s
370 END REPeat main
```

EUROPA-AMÉRICA

Convidam-no a juntar-se
àquele grupo de leitores exigentes
para quem um livro,
além de uma companhia agradável,
é uma companhia útil.
Assim, teremos muito gosto
em passar a enviar-lhe, regularmente,
informações sobre os livros
que publicamos.

o Editor

Nome: ______________________

Profissão: ______________________

Morada: ______________________ *Tel.:* __________

Cód. Postal: __________ *Localidade:* ______________________

Encontrei este postal no livro: ______________________

Que adquiri numa livraria *Que encomendei pelo correio*

NÓS EDITAMOS O LIVRO QUE VOCÊ PROCURA!

IMPORTANTE: Se desejar alguns livros de nossa edição, também poderá usar este postal. Bastará preencher o espaço que se segue. Todas as encomendas são confidenciais.

Referência *Título e Autor*

A utilizar somente no Continente e Regiões Autónomas

RSF

IMPRESSO — RESPOSTA
Autorizado pelos CTT

Não carece de selo.
O porte será pago pelo destinatário.

PUBLICAÇÕES EUROPA-AMÉRICA
Apartado 8
2726 MEM MARTINS CODEX